

This page Is Inserted by IFW Operations
And is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of
The original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
Please do not report the images to the
Image Problem Mailbox.

【書類名】 特許願
【整理番号】 9805670
【提出日】 平成11年 5月19日
【あて先】 特許庁長官殿
【国際特許分類】 G06F 17/50
F16H 21/00
【発明の名称】 機構制御プログラム開発支援システム、機構制御プログラム開発支援装置、および機構制御プログラム開発支援プログラム記憶媒体
【請求項の数】 3
【発明者】
【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通
株式会社内
【氏名】 千田 陽介
【発明者】
【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通
株式会社内
【氏名】 佐藤 裕一
【発明者】
【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通
株式会社内
【氏名】 平田 光徳
【特許出願人】
【識別番号】 000005223
【氏名又は名称】 富士通株式会社
【代理人】
【識別番号】 100094330
【弁理士】
【氏名又は名称】 山田 正紀
【先の出願に基づく優先権主張】
【国際出願番号】 PCT/JP/98/05685
【出願日】 平成10年12月16日
【手数料の表示】
【予納台帳番号】 017961
【納付金額】 21,000円
【提出物件の目録】
【物件名】 明細書 1
【物件名】 図面 1
【物件名】 要約書 1
【包括委任状番号】 9704376
【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 機構制御プログラム開発支援システム、機構制御プログラム開発支援装置、および機構制御プログラム開発支援プログラム記憶媒体

【特許請求の範囲】

【請求項 1】 内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータと

前記三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、前記三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行する制御プログラムプロセッサとを備えたことを特徴とする機構制御プログラム開発支援システム。

【請求項 2】 内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、前記三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行することを特徴とする機構制御プログラム開発支援装置。

【請求項 3】 アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレーションプログラムと、

前記三次元機構モデルの動作を制御する制御プログラムに付加されて、該制御プログラムを、三次元機構モデルの動作との同期をとりながら実行させる同期化プログラムとを有する機構制御プログラム開発支援プログラムが記憶されてなることを特徴とする機構制御プログラム開発支援プログラム記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、機構を制御する制御プログラムの開発を支援する機構制御プログラム開発支援システムおよび機構制御プログラム開発支援装置、および機構を制御する制御プログラムの開発を支援する機構制御プログラム開発支援プログラムを記憶してなる機構制御プログラム開発支援プログラム記憶媒体に関する。

【0002】

【従来の技術】

近年、コンピュータグラフィックスとシミュレーション技術の発展で、コンピュータ内に三次元機構モデルを構築し、製品を試作することなくシミュレーションによってその三次元機構モデルの可動部を動かし、部品どうしが不用意に接触せず所期の動作を行なうことができるかどうか等、その機構の確認を行なうことができるようになってきており、その結果として、設計の不具合を早期に発見し、部品の配置変更や可動部を考慮した修正等を行なうことが可能になり、製品開発に要する時間の短縮化やコスト削減を図ることができるようになってきている。

【0003】

ところで、そのような機構を動作させるには、そのような機構を制御する制御プログラムが開発され、その制御プログラムを実行することにより機構を動作させるのが一般的である。

【0004】

ところが従来は、そのような制御プログラムを開発するには、例えば三次元機構モデルの動作シミュレーション等により動作上ほぼ完成した機構を物理的に試作し、その試作した機構を動作させながら制御プログラムの開発が行なわれている。このように、制御プログラムの開発、デバッグには試作品が必要であるため、試作品が完成するまでの間や試作品を修理、改造している間は制御プログラムの開発を行なうことができず、制御プログラムの開発に時間やコストがかかっていた。

【0005】

【発明が解決しようとする課題】

本発明は、上記事情に鑑み、機構を制御する制御プログラムを容易に開発することができるよう支援する機構制御プログラム開発支援システムおよび機構制御プログラム開発支援装置、および機構を制御する制御プログラムの開発を容易化する機構制御プログラム開発支援プログラムを記憶してなる機構制御プログラム開発支援プログラム記憶媒体を提供することを目的とする。

【0006】

【課題を解決するための手段】

上記目的を達成する本発明の機構制御プログラム開発支援システムは、内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータと、

三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行する制御プログラムプロセッサとを備えたことを特徴とする。

【0007】

三次元機構モデルを構築しそれをシミュレーションで動作させる場合の動作速度は、その三次元機構モデルに相当する製品を試作し、その試作品を動作させる場合の動作速度とは一般には大きく異なっており、このままでは制御プログラムを実行させて三次元機構モデルを動作させても制御プログラムのデバックには不適当である。これに対し、本発明の機構制御プログラム開発支援システムを構成する制御プログラムプロセッサは、制御プログラムを、その制御プログラムにより制御される三次元機構モデルの動作との同期をとりながら実行するものであるため、三次元機構モデルを制御プログラムで制御しながら動作させてその動作を確認することによりその制御プログラムのデバッグを行なうことができる。

【0008】

このように、本発明の機構制御プログラム開発支援システムによれば、試作品を製作することなく、三次元機構モデルをシミュレーションで動作させながら動作プログラムのデバッグを行なうことができるため、動作プログラムの開発に要する時間を短縮化し、開発コストを大きく低減することができる。

【0009】

ここで、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルの動作と制御プログラムの実行との同期をとるにあたっては、

三次元機構モデルシミュレータが、制御プログラムプロセッサ内で動作する制御プログラムの実行時間を制御プログラムプロセッサに通知し、

制御プログラムプロセッサが、三次元機構モデルシミュレータからの制御プログラムの実行時間の通知を受けその実行時間だけ制御プログラムを実行して、その実行時間分の制御プログラムの実行の終了を三次元機構モデルシミュレータに通知し、

三次元機構モデルシミュレータが、制御プログラムプロセッサからの制御プログラムの実行時間分の実行の終了の通知を受け、三次元機構モデルを、制御プログラムのその実行時間分の制御に応じた動作量だけ動作させて、制御プログラムの次の実行時間を決定するものであってもよい。

【0010】

例えばこのような手順を踏むことにより、三次元機構モデルの動作と制御プログラムの実行との同期をとることができる。

【0011】

ここで、上記の手順を採用して同期をとるにあたっては、三次元機構モデルシミュレータが、制御プログラムプロセッサ内で動作する制御プログラムの実行時間を、三次元機構モデルを構成する部品どうしの間の距離に応じて決定するものであることが好ましい。

例えば部品どうしが大きく離れているときは、実行時間を大きくとって一回に

大きく動作させることにより所期の全体の動作を行なわせるまでの時間を短縮することができ、一方部品どうしが近づいたときは、実行時間を小さい値として少しづつ動かすことにより正確な動作確認を行なうことができる。

【0012】

また、部品間の距離を求めるにあたっては、三次元機構モデルシミュレータが、動作プログラムの実行と連携した三次元機構モデルの動作に先立って、三次元機構モデルを構成する複数の部品を、三次元機構モデルを構成するアクチュエータごとに、そのアクチュエータの動作に応じて動作する部品の集合と、そのアクチュエータの動作によっては静止した状態にとどまる部品の集合とに分離し、動作プログラムの実行と連携した三次元機構モデルの動作にあたっては、これら分離した部品の集合どうしの間の、これら部品の集合どうしの間で常に干渉した状態にある部品間の距離を除く距離を求めるものであることが好ましい。

【0013】

上記のように部品の集合に分け、それら部品の集合どうしの間の距離を求ることにより、距離を求めるための演算時間を大幅に短縮化することができる。

【0014】

この場合に、さらに、三次元機構モデルシミュレータが、動作プログラムの実行と連携した三次元機構モデルの動作に先立って、その三次元機構モデルを構成するアクチュエータごとに、三次元機構モデルを構成する複数の部品のうちの、アクチュエータの動作に応じて動作する部品の集合を抽出し、そのアクチュエータを動作させることによりその集合を構成する部品を動作させてそれらの部品どうしの干渉の発生を検出し、動作プログラムの実行と連携した三次元機構モデルの動作にあたっては、常に干渉した状態にある部品を除く干渉が発生した部品どうしの間の距離を求めるものであることが好ましい。

【0015】

上記のように部品の集合に分けたとき、アクチュエータの動作に応じて動作する部品の集合の中でも部品どうしが互いに干渉する可能性がある。このような場合に、その集合を構成する部品を動作させてそれらの部品どうしの干渉の発生を検出し、干渉の発生した部品どうしの間の距離を求ることにより、干渉の可能性のある部品どうしの間の距離を洩れなく求めることができる。

【0016】

さらに、三次元機構モデルシミュレータが、三次元機構モデルを構成する部品間の距離を、その三次元機構モデルを構成するアクチュエータが動作している間のみ求めるものであることが好ましい。

【0017】

距離を求める演算を必要な時のみに限ることにより、シミュレーションの高速化が図られる。

【0018】

また、三次元機構モデルシミュレータ内で動作する三次元機構モデルと動作プログラムプロセッサ内で実行される動作プログラムとの同期をとるにあたっては、制御プログラムプロセッサ内で実行される制御プログラムが、少なくとも一部に、所定周期で繰り返し実行される部分を有し、その部分の実行回数に基づいて、その制御プログラムの実行時間分の実行の終了を認識するものであることが好ましい。

【0019】

制御プログラムは、例えば一定時間ごとのタイマ割り込みで実行されるなど、所定周期で繰り返し実行される部分を有するのが一般的である。そこでこのような場合に、その部分が実行された回数を計数することによって制御プログラムの実行時間分の実行を認識することができる。

【0020】

また、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するアクチュエータを定

義するアクチュエータ定義手段を備え、該アクチュエータ定義手段が、該アクチュエータの定義にあたり、速度指令値と目標速度とを1:1で定義する有段変速と、速度指令値と目標速度との間の比例定数を定義する無段変速との双方の定義メニューを用意してなるものであることが好ましい。

【0021】

このような2つの定義方法の双方を採用することにより、アクチュエータの定義の自由度が向上する。尚、ここでは‘有段変速’と‘無段変速’との2種類について述べたが、それらの‘有段変速’あるいは‘無段変速’の中でさらに複数種類の定義方法を採用してもよい。後述する実施形態では‘無段変速’の定義方法の中にさらに2種類の定義方法が採用されている。

【0022】

また、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、その三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するアクチュエータを一次遅れ系として動作させるものであることが好ましい。

【0023】

アクチュエータを一次遅れ系として動作させることにより、そのアクチュエータの動作がほぼ一定の動作速度に達するまでの遅れ時間をあらわす整定時間を指定するだけでそのアクチュエータの立ち上がり時の振舞いを規定することができ、アクチュエータの動作の定義が容易となる。

【0024】

さらに、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するアクチュエータを定義するアクチュエータ定義手段を備え、そのアクチュエータ定義手段が、アクチュエータの動作により最初に動作する部品を、三次元機構モデルを三次元機構モデルシミュレータ内で動作させる際のアクチュエータとして定義するものであることが好ましい。

【0025】

アクチュエータをこのように定義することで、三次元機構モデルの動作シミュレーションが容易となる。

【0026】

さらに、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサにより検出される部品を、三次元機構モデルを該三次元機構モデルシミュレータ内で動作させる際のセンサとして定義するものであることが好ましい。

【0027】

センサをこのように定義すると、上記のアクチュエータの定義と同様、シミュレーションが容易となる。

【0028】

さらに、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、複数の部品間の干渉を検出することによりオン、オフを判定するセンサを定義するものであってもよい。

【0029】

センサは、このような定義方法によってもシミュレーションを容易化することができる。

また、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するエンコーダを定義するエンコーダ定義手段を備え、三次元機構モデルシミュレータが、制御プログラムプロセッサからの制御プログラムの上記実行時間の実行の終了の通知を受け、三次元機構モデルを、制御プログラムのその実行時間分の制御に応じた動作

量だけ動作させた段階で、その実行時間分の制御に応じたエンコーダの動作量に応じたパルス数のパルスを出力するものであることが好ましい。

【0030】

エンコーダを定義し、1回の実行時間分の制御量だけ動作させた段階でその実行時間分の制御に応じたエンコーダの動作量に応じたパルス数のパルスを出力することにより、エンコーダを正しくシミュレートすることができる。

【0031】

ここで、上記エンコーダ定義手段が、エンコーダの出力パルスのパルス周波数の上限を設定する機能を有し、上記三次元機構モデルシミュレータが、エンコーダの動作量に応じたパルス数のパルスを出力するにあたり、設定されたパルス周波数の上限以下のパルス周波数のパルスを出力するものであることが好ましい。

【0032】

現実のエンコーダの出力パルスはある一定のパルス周波数以下の周波数を持っており、パルス周波数の上限を設定しその上限以下のパルス周波数のパルスを出力することにより、より正確なシミュレートが可能となる。

【0033】

また、本発明の機構制御プログラム開発支援システムにおいて、上記三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するポテンショメータを定義するポテンショメータ定義手段を備えたものであることが好ましい。

【0034】

ポテンショメータも三次元機構モデルの1つの重要な構成要素となり得るからである。

【0035】

また、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサの状態を変化させる原因の発生からセンサの状態の実際の変化までの遅れを定義する手段を含むものであることが好ましい。

【0036】

制御プログラムの開発にあたっては、アクチュエータやセンサが異常動作したときにも、暴走せずにあるコントロールされた状態を保つことが好ましく、そのような制御プログラムの開発にあたっては、アクチュエータやセンサの異常動作をシミュレートする必要がある。

本発明の機構制御プログラム開発支援システムがセンサ定義手段を備え、そのセンサ定義手段が、センサの状態をオンあるいはオフからオフあるいはオンに変化させる原因の発生からそのセンサの状態の実際の変化までの遅れを定義する手段を含むものであると、その遅れを定義し、センサの応答遅れという異常をシミュレートすることができる。

【0037】

また、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサの状態の変化に伴うチャッタリングを定義する手段を含むものであることが好ましい。

【0038】

チャッタリングも、センサの、よく発生する異常動作の1つであり、このチャッタリングをシミュレートすることも重要である。

【0039】

さらに、上記本発明の機構制御プログラム開発支援システムは、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサの故障を定義する手段を含むものであることが好ましく、また、上記本発明の機構制御プログラム開発支援システムは、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構

成するアクチュエータを定義するアクチュエータ定義手段を備え、そのアクチュエータ定義手段が、アクチュエータの故障を定義する手段を含むものであることが好ましい。

【0040】

センサやアクチュエータの故障をシミュレートすることも重要である。

【0041】

三次元機構モデルは数学的なモデルであるから、2つの部品どうしが重なっていても自由に動かすことができるが、実際の部品の場合は部品どうしが干渉するとそれ以上動作させることができない事態が生じる。したがって、上記本発明の機構制御プログラム開発支援システムにおいて、上記三次元機構モデルシミュレータは、三次元機構モデルを動作させている途中でその三次元機構モデルを構成する部品どうしの間に干渉が生じた場合に、干渉が生じた部品の動作を停止するものであることが好ましい。

【0042】

また上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成する部品のうちのいずれかの部品を指定してその部品の動作を禁止する不良部品定義手段を備えることが好ましい。

【0043】

センサやアクチュエータのみでなく、通常の従動部品も故障して動作不良となることもあります、それをシミュレートすることも重要である。

【0044】

その場合に、上記三次元機構モデルシミュレータが、三次元機構モデルを動作させている途中における上記不良部品定義手段により指定された部品の指定を受けて、指定された部品の現在の動作方向への動作を禁止するものであることが好ましく、上記三次元機構モデルシミュレータが、上記不良部品定義手段による部品の、動作が禁止された方向とは逆の方向への動作に応じて、その部品の、動作が禁止された方向についての動作の禁止を解除するものであることが好ましい。

【0045】

このようにして、何かに引っ掛かって動作不良となり、逆方向に動かすことにより正常に戻る状態をシミュレートすることができる。

【0046】

また、上記本発明の機構制御プログラム開発支援システムにおいて、上記三次元機構モデルシミュレータが、三次元機構モデルを構成する部品の可動範囲を、その部品を動作させたときのその部品と他の部品との間の干渉を検出することにより自動設定する可動範囲設定手段を有するものであることが好ましい。

【0047】

干渉検出の手法を用いて部品の可動範囲を自動設定することにより、その可動範囲をオペレータがいちいち入力する手間が省かれ、使い勝手の良い、操作性に優れたシステムが実現する。

【0048】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、三次元機構モデルを構成する2つの部品のカム関係を、これら2つの部品のうちの一方の部品を動作させたときのこれら2つの部品の間の干渉を検出することにより自動設定するカム関係設定手段を有するものであることが好ましい。

【0049】

カム関係にある2つの部品の連携動作に関しても、干渉検出の手法により自動設定することにより、オペレータの手間が省かれ、使い勝手の良いシステムが実現する。

【0050】

カム関係を自動設定するにあたっては、2つの部品のうちの一方がバネや重力

により一方向に付勢されたものである場合がある。その場合、カム関係設定手段が、上記2つの部品のうちの他方の部品が所定方向に付勢されたものである場合の、これら2つの部品の間の干渉を検出することにより、これら2つの部品のカム関係を自動設定するものであることが好ましい。

【0051】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、三次元機構モデルを構成する2つの部品のうちの一方の部品の溝に他方の部品のピンが挿入された溝関係を、そのピンに代えてそのピンよりも細径の仮想部品を作成しその仮想部品と溝壁との間の距離を検出することにより自動設定する溝関係設定手段を有するものであることが好ましい。

【0052】

溝に挿入されたピンは、通常その溝壁に常に接触しており、このままでは非接触状態を作り出すことができない。そこで、上記のような細径の仮想部品を作成し、溝壁との間の短縮を検出することにより、溝関係を自動設定することができる。

【0053】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、三次元機構モデルを構成する部品を動作させるプログラムを再帰的に実行することにより三次元機構モデルを動作させるものであることが好ましい。

【0054】

このような再帰的に実行されるプログラムを採用することにより、三次元機構モデルの動作シミュレーションの高速化が可能になる。

【0055】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成する2つの部品の連携動作の有無を切り換えるクラッチを定義するクラッチ定義手段を備えたものであることが好ましい。

【0056】

クラッチも、三次元機構モデルを構成する重要な部品の1つとなり得るため、これをシミュレートすることができるようにしておくことが好ましい。

【0057】

また、上記目的を達成する本発明の機構制御プログラム開発支援装置は、内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行するものであることを特徴とする。

【0058】

三次元機構モデルシミュレータが別に構成されているときは、機構制御プログラム開発支援装置を構成することにより、動作プログラムの開発支援を行なうことができる。

【0059】

また、上記目的を達成する本発明の機構制御プログラム開発支援プログラム記憶媒体は、

アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレーションプログラムと、

三次元機構モデルの動作を制御する制御プログラムに付加されて、その制御プログラムを、三次元機構モデルの動作との同期をとりながら実行させる同期化プログラムとを有する機構制御プログラム開発支援プログラムが記憶されてなることを特徴とする。

【0060】

本発明の機構制御プログラム開発支援プログラム記憶媒体に記憶された機構制御プログラム開発支援プログラムをコンピュータ内で実行することにより、シミュレーション上で動作プログラムのデバッグを行なうことができ、動作プログラムの開発が容易となる。

【0061】

ここで、上記本発明の機構制御プログラム開発支援プログラム記憶媒体において、

三次元機構モデルシミュレートプログラムが、制御プログラムの実行時間を同期化プログラムに通知し、

同期化プログラムが、三次元機構モデルシミュレートプログラムからの、制御プログラムの実行時間の通知を受け、その実行時間分だけ制御プログラムを実行させて、その実行時間分の制御プログラムの実行の終了を三次元機構モデルシミュレートプログラムに通知し、

三次元機構モデルシミュレートプログラムが、同期化プログラムからの、制御プログラムの上記実行時間分の実行の終了の通知を受け、三次元機構モデルを、制御プログラムのその実行時間分の制御に応じた動作量だけ動作させて、制御プログラムの次の実行時間を決定するものであることが好ましい。

【0062】

たとえばこのような通知のやりとりにより、三次元機構モデルの動作と制御プログラムの実行との間で同期をとることができる。

【0063】

【発明の実施の形態】

以下、本発明の実施形態について説明する。

【0064】

図1は、内部に機構制御プログラム開発支援システムが構築されたコンピュータシステムの外観図である。

【0065】

このコンピュータシステム100は、CPU、RAM、ハードディスク等を内蔵した本体部101、本体部101からの指示により表示画面102aに画面表示を行うCRTディスプレイ102、このコンピュータシステムにユーザの指示や文字情報を入力するためのキーボード103、表示画面102a上の任意の位置を指定することによりその位置に表示されていたアイコン等に応じた指示を入力するマウス104を備えている。

【0066】

本体部101は、さらに、外観上、フロッピィディスク212（図1には図示せず；図2参照）やCDROM210が取り出し自在に装填されるフロッピィディスク装填口101aおよびCDROM装填口101bを有しており、その内部には、装填されたフロッピィディスクやCDROM210をドライブする、フロッピィディスクドライブ224、CDROMドライブ225（図10参照）も内蔵されている。

【0067】

ここでは、CDROM210に、本発明にいう機構制御プログラム開発支援プログラムが記憶されている。このCDROM210がCDROM装填口101bから本体部101内に装填され、CDROMドライブ225により、そのCDROM210に記憶された機構制御プログラム開発支援プログラムがこのコンピュータシステム100のハードディスク内にインストールされる。さらに、このコンピュータシステム100のハードディスクには、ユーザによりキーボードやマウス等を用いて定義された三次元機構モデル、および、ユーザにより、やはりキーボードやマウス等を用いて入力された、その三次元機構モデルを実際に製造したときの製品を制御するための制御プログラムも格納されている。尚、三次元機構モデルや制御プログラムは、それらのうちのいずれか一方あるいは双方が、この図1に示すコンピュータ100とは別のシステムで構築ないしプログラミング

さて、このコンピュータ100に格納されたものであってもよい。

【0068】

このコンピュータシステム100内に構築された三次元機構モデルは、CDR0M210からこのコンピュータシステム100内にローディングされた機構制御プログラム開発支援プログラムを構成する三次元機構モデルシミュレーションプログラムによりそのコンピュータシステム100内で動作し、一方、このコンピュータシステム100内に組み込まれた動作プログラムは、CDROM210からこのコンピュータシステム100内にローディングされた機構制御プログラム開発支援プログラムを構成する同期化プログラムと結合され、このコンピュータシステム100内では、その動作プログラムが、三次元機構モデルの動作と同期して実行される。

【0069】

ここでは、動作プログラムのデバッグを対象としており、したがってここでは、コンピュータシステム100内に構築された三次元機構モデル自体は、動作プログラムとの連携動作の途中で不具合が発見されたときは修正されることがあるもの一応その動作が確認されたものであり、動作プログラムはデバッグ中のものであるとする。

【0070】

ここでは、本発明との対応関係でいうと、このコンピュータシステム100の、三次元機構モデルを動作させる機能が三次元機構モデルシミュレータに相当し、同期化プログラムと結合された動作プログラムを実行する機能が制御プログラムプロセッサおよび機構制御プログラム開発支援装置に相当する。

【0071】

このように、本発明の機構制御プログラム開発支援システムには、1つの装置（ここではコンピュータシステム100）内に三次元機構モデルシミュレータと動作プログラムプロセッサとの双方が組み込まれた形態も含まれる。

【0072】

図2は、図1に外観を示すコンピュータシステムのハードウェア構成図である。

【0073】

ここには、中央演算処理装置（CPU）221、RAM222、ハードディスクコントローラ223、フロッピィディスクドライブ224、CDROMドライブ225、マウスコントローラ226、キーボードコントローラ227、およびディスプレイコントローラ228が示されており、それらはバス220で相互に接続されている。

【0074】

フロッピィディスクドライブ224、CDROMドライブ225は、図1を参照して説明したように、フロッピィディスク212、CDROM210が装填され、装填されたフロッピィディスク212、CDROM210をアクセスするものである。

【0075】

また、ここには、ハードディスクコントローラ223によりアクセスされるハードディスク211、マウスコントローラ226により制御されるマウス104、キーボードコントローラ227により制御されるキーボード103、およびディスプレイコントローラ228により制御されるCRTディスプレイ102も示されている。

【0076】

前述したように、CDROM210には機構制御プログラム開発支援プログラムが記憶されており、CDROMドライブ225により、そのCDROM210からその機構制御プログラム開発支援プログラムが読み込まれ、バス220を経由し、ハードディスクコントローラ223によりハードディスク211内に格納される。実際の実行にあたっては、そのハードディスク211内のプログラムは

RAM222上にロードされ、CPU221により実行される。

【0077】

図3は、プログラム記憶媒体内に記憶されたプログラムの構成を示す図である

【0078】

この図3に示すプログラム記憶媒体300は、機構制御プログラム開発支援プログラムが記憶されてなるCD-ROM210、そのプログラムがインストールされた状態のハードディスク211、およびそのプログラムがフロッピィディスクにダウンロードされたときの、そのダウンロードされたプログラムを記憶した状態にあるフロッピィディスク212等を代表的に示したものである。

【0079】

この図3に示すプログラム記憶媒体300には機構制御プログラム開発支援プログラム310が記憶されている。この機構制御プログラム開発支援プログラムは、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレーションプログラム311と、その三次元機構モデルを制御する制御プログラムに付加される同期化プログラム312とから構成されている。この同期化プログラムは、制御プログラムに付加されて、その制御プログラムを、三次元機構モデルシミュレーションプログラム311の実行による三次元機構モデルの動作との同期をとりながら実行させるプログラムである。

【0080】

機構制御プログラム開発支援プログラム310を構成する、三次元機構モデルシミュレーションプログラム311と同期化プログラム312は、図1に示すコンピュータシステム内で実行される際、相互に以下のようないくつかの通信を行なうことにより、三次元機構モデルの動作と制御プログラム実行との同期がとられる。すなわち、三次元機構モデルシミュレーションプログラム311が、制御プログラムの実行時間を同期化プログラム312に通知する。同期化プログラム312は、その通知を受けて、その通知を受けた実行時間分だけ制御プログラムの実行を許容し、制御プログラムがその実行時間だけ実行されると制御プログラムの実行をその段階で中断させ、制御プログラムのその実行時間分だけの実行が終了したことを三次元機構モデルシミュレーションプログラム311に通知する。三次元機構モデルシミュレーションプログラム311は、その同期化プログラム312からの通知を受けて、三次元機構モデルを、動作プログラムの、その実行時間分の制御に応じた動作量だけ動作させて、制御プログラムの次の実行時間を決定し、その決定した実行時間を同期化プログラム312に通知する。これを繰り返すことにより、相互に同期がとられた形態で、動作プログラムが実行され、その動作プログラムの実行による制御に従って三次元機構モデルが動作する。

【0081】

図4は、本発明の機構制御プログラム開発支援システムの一例が構築されたコンピュータネットワークの一例を示す図である。

【0082】

図4には、2台のコンピュータシステム400、500が通信ネットワーク600を介して接続されている様子が示されており、ここでは、コンピュータシステム400が本発明にいう三次元機構モデルシミュレータを構成し、コンピュータシステム500が本発明にいう制御プログラムプロセッサを構成している。

【0083】

各コンピュータシステム400、500は、CPU、RAM、ハードディスク、通信用モデム等を内蔵した本体部401、501、本体部401、501からの指示により表示画面402a、502aに画面表示を行うCRTディスプレイ402、502、ユーザの指示や文字情報を入力するためのキーボード403、503、表示画面402a、502a上の任意の位置を指定することによりその位置に表示されていたアイコン等に応じた指示を入力するマウス404、504

を備えている。

【0084】

本体部401, 501には、外観上、フロッピィディスクやCDROMが取り出し自在に装填されるフロッピィディスク装填口401a, 501aやCDROM装填口401b, 501bを有しており、その内部には、装填されたフロッピィディスク、CDROMをドライブするフロッピィディスクドライバ、CDROMドライバも内蔵されている。

【0085】

ここでは、機構制御プログラム開発支援プログラムのうちの三次元機構モデルシミュレーションプログラム311(図3参照)は、コンピュータシステム400にインストールされ、同期化プログラム312は、コンピュータシステム500にインストールされる。コンピュータシステム400内には三次元機構モデルが構築されており、コンピュータシステム400内にインストールされた三次元機構モデルシミュレーションプログラム311は、そのコンピュータシステム400内に構築された三次元機構モデルを動作させる。一方、コンピュータシステム500には、コンピュータシステム400内に構築された三次元機構モデルを実際に製造したときの製品を制御するための制御プログラムも格納されており、このコンピュータシステム500にインストールされた同期化プログラムは制御プログラムと統合され、コンピュータシステム500内で動作する三次元機構モデルの動作を制御する。三次元機構モデルシミュレーションプログラム311と同期化プログラム312との間の通信は、通信ネットワーク600を経由して行なわれる。

【0086】

本発明の機構制御プログラム開発支援システムには、この図4に示すように、三次元機構モデルシミュレータと制御プログラムプロセッサがそれぞれ別々の装置(ここではコンピュータシステム400とコンピュータシステム500)に組み込まれ、それらの間で通信を行なうように構成された形態も含まれる。

【0087】

また、本発明の機構制御プログラム開発支援システムには、以下のような形態も存在する。すなわち電源や水晶を繋げたマイコンチップに制御プログラムをローディングし、例えば図4におけるコンピュータシステム400にI/Oボードを挿入してマイコンチップと接続し、マイコンチップから出力された、本来はモータドライバに伝達される電気信号をI/Oボードを介してコンピュータシステム400に入力し、コンピュータシステム400では三次元機構モデルを動作させて本来センサから出力される信号と同じ信号をI/Oボードを介して出力し、その信号をマイコンチップで受信する。

【0088】

このようにして、三次元機構モデルはコンピュータシステム400内で動作させ、制御プログラムは実際の使用形態に近いマイコンチップで動作させるという環境を実現してもよい。

【0089】

図5は、図4に外観を示すコンピュータネットワークを構成するコンピュータシステムのハードウェア構成図である。

【0090】

図4に示すコンピュータシステム400, 500は、本実施形態ではハードウェア構成は同一であり、ここでは、コンピュータシステム400を例に挙げて説明する。

【0091】

ここには、中央演算処理装置(CPU)421、RAM422、ハードディスクコントローラ423、フロッピィディスクドライバ424、CDROMドライバ425、マウスコントローラ426、キーボードコントローラ427、およびディスプレイコントローラ428が示されており、それらはバス420で相互に

接続されている。

【0092】

またここには、ハードディスク411、フロッピィディスク412、CD-ROM410、マウス404、キーボード403、CRTディスプレイ402も示されている。これらの各要素は、図2に示したハードウェア構成図における対応する各構成要素とそれぞれ同じ作用を成すものであり、詳細説明は省略する。

【0093】

図2には示されていない要素として、この図5には、モデム429が示されている。このモデム429は、図4に示す通信ネットワーク600に接続され、2台のコンピュータシステム400、500間での通信を担当するものである。

【0094】

以上では、図1および図2と、図3および図4との2種類のハードウェア構成を示したが、以下ではそれらのコンピュータシステムあるいはコンピュータネットワーク内に構築された機構制御プログラム開発支援システムの具体的な内容について説明する。

【0095】

図6は、機構制御プログラム開発支援システムの概念図である。

【0096】

三次元機構モデルシミュレータ（「3Dモデルシミュレータ」と略記する）内には三次元機構モデル（「3Dモデル」と略記する）が構築されており、一方、制御プログラムプロセッサ内には、その3Dモデルを制御する制御プログラムがインストールされている。

【0097】

この制御プログラムプロセッサ内の制御プログラムは、3Dモデルに相当する製品を実際に製作したときの、その製品を制御する制御プログラムに、3Dモデルシミュレータ内で動作する3Dモデルの動作との同期をとるための同期化プログラムとが結合したものである。

また3Dモデルは、複数のリンク（製品を構成する個々の部品）の三次元形状とそれらのリンクの姿勢の関係を定義した三次元機構データによって定義される、製品の三次元モデルであり、リンクの姿勢を変更することによってその3Dモデルが動作することになる。

【0098】

制御プログラムプロセッサから3Dモデルシミュレータにはモータ等のアクチュエータを動作させるためのアクチュエータ指令信号が送られ、3Dモデルシミュレータから制御プログラムプロセッサには、センサのオン、オフ等をあらわすセンサ信号が送られる。また、これら制御プログラムプロセッサと3Dモデルシミュレータとの間では同期信号が送受信される。

【0099】

図7は、センサやモータの定義時の処理を示す3Dモデルシミュレータの模式図、図8は、シミュレーション時の3Dモデルシミュレータの処理を示す模式図である。

【0100】

3Dモデルは、三次元CAD等を用いて構築され、その構築された3Dモデルに対し、ここでは、図7に示すように、ユーザによりキーボード等の入力装置が操作されてモータやセンサが定義される。そのような定義が行なわれた後の実際のシミュレーションにあたっては、制御プログラムプロセッサから送られてきたアクチュエータ指令信号（ここではモータ回転指令信号）に対応し、モータとして定義されたリンク（以下、モータとして定義されたリンクを「モータリンク」と称する）の姿勢を変化させ、さらに機構データに基づき、モータリンクに連結された他のリンクについて、モータリンクの姿勢変化に応じた量だけ動作させ、その結果として、センサとして定義されたリンク（以下、センサとして定義されたリンクを「センサリンク」と称する）の姿勢にも影響が出る。そこでセンサリ

ンクの姿勢に応じセンサ信号を発生させ、そのセンサ信号は制御プログラムプロセッサに送信される。

【0101】

図9は、シミュレーション時における3Dモデルシミュレータと制御プログラムの動作タイミングチャートを示す図である。

【0102】

図中のハッチング部分は、シミュレータ、制御プログラムがそれぞれ動作していることを示している。図示のように、シミュレータと制御プログラムは動作、停止のサイクルを交互に繰り返している。各サイクルは次のような手順で動作する。ただし、これらシミュレータと制御プログラムとの間の同期に関する詳細な説明は後にゆずる。

【0103】

(1) シミュレータは、入力アクチュエータ信号を基に、シミュレーション時間 Δt 分のシミュレーションを行ない、センサ信号を出力する。

【0104】

(2) シミュレータは、現在の3Dモデルの干渉可能性から、次のループでのシミュレーション時間 Δt を決定し、その時間 Δt を制御プログラムに受け渡すとともに、動作開始指令 (Ready Flag) を制御プログラムに受け渡す。

【0105】

(3) 制御プログラムは時間 Δt を受け取るとともに動作開始指令 (Ready Flag) を受け取って、その時間 Δt 分の計算 (制御) を行なった後、シミュレータ側に動作終了信号 (Loop Flag) を送り、次の動作開始指令 (Ready Flag) を受け取るまでの間、動作を停止する。

【0106】

(4) シミュレータは制御プログラムの動作終了信号 (Loop Flag) を受け取って時間 Δt 分のシミュレーションを開始する。

【0107】

シミュレータと制御プログラムは、以上を繰り返すことにより交互に動作する。

【0108】

図10は、シミュレーション時のシミュレータのフローを示すフローチャートである。

【0109】

シミュレーションが開始されると、先ず初期設定が行なわれて (ステップS101)、3Dモデルの各リンクの姿勢が初期化される。次いで、シミュレーションを行なう時間間隔 (シミュレーション間隔) Δt が制御プログラム側に送信され (ステップS102)、制御プログラム側からシミュレーション間隔 Δt を受信した旨をあらわす信号を受け取ると (ステップS103)、制御プログラムに制御プログラム計算開始命令を送信する (ステップS104)。制御プログラムによりシミュレーション間隔 Δt 分の計算が行なわれてその終了が通知されると (ステップS105)、ここでシミュレートしている3Dモデルを構成する全てのモータおよび全てのセンサを今回の Δt 分の処理に関し未処理である旨初期設定される (ステップS106)。

【0110】

次いでステップS201では、未処理のモータが存在しているか否か判定され、未処理のモータ1つずつについてステップS202～S207の処理が行なわれる。すなわち、制御プログラム側から送られてきた、今処理しようとしているモータに関するアクチュエータ信号に応じてそのモータの回転速度が決定され (ステップS202)、その回転速度から Δt 間に変位するモータの姿勢が決定される (ステップS203)。次いで機構関係定義によりそのモータに関連づけられた他の部品の姿勢が変更され (ステップS204)、部品間の最短距離 d が測

定される（ステップS205）。ステップS206、S207では、各モータの回転により測定される各最短距離dの中の最短距離Dが求められる。このようにして、全てのモータに関し、 Δt の間の姿勢変更が終了すると、姿勢変更後の3Dモデルが画面に表示される（ステップS208）。

【0111】

次いで、今度は未処理のセンサが存在しているか否か判定され（ステップS301）、未処理のセンサ1つずつについてセンサリンクの姿勢に応じたセンサ信号のオン、オフが検出される（ステップS302）。全てのセンサについての処理が終了すると、制御プログラムに向けてそれら全てのセンサに関するセンサ信号が出力され（ステップS303）、さらに部品間最短距離Dに応じたサンプリング間隔 Δt が決定され（ステップS401）、ステップS102に戻ってその決定された間隔 Δt が制御プログラムに送信される。

【0112】

シミュレータでは、以上の処理が繰り返し実行され、制御プログラムとの同期をとりつつ、3Dモデルの動作シミュレーションが行なわれる。

【0113】

図11は、図10に示すシミュレータの処理フローの一部分（図10に一点鎖線で囲った部分）に代えて採用することのできる部分フローを示すフローチャートである。

【0114】

この図11には、ステップS202とステップS203との間にステップS210が挿入されている。このステップS210では、今処理を行なっているモータの回転速度がゼロ（ $\omega=0$ ）か否かが判定され、今回の間隔 Δt の間にそのモータは回転しない（ $\omega=0$ ）のときはそのモータの動作に関しては部品間の距離dの測定は行なわずにステップS211に進んでd=∞とし、ステップS206の判定でその距離d=∞が距離Dを求める演算から除外されるようにしている。

【0115】

部品間の距離を求めるにはかなりの計算量を要し、したがってモータが回転しないときはそのモータに関する距離の測定を行なわないようにすることにより、計算量を減らし、シミュレーションのスピードアップを図ることができる。

【0116】

図12は、モータとして定義されるリンクの選び方を示す模式図である。

【0117】

この図12において、本来のモータは図中のリンクAであるが、リンクA自体は土台に固定されているため、シミュレーション上は、その本来のモータによって一番最初に駆動されて3Dモデル上姿勢が変更されるリンクBがモータとして定義される。モータをこのように定義することでシミュレーションが容易となる。

【0118】

図13、図14は、センサとして定義されるリンクの選び方を示す模式図である。

【0119】

これら図13、図14において、本来のセンサはリンクAであるが、ここでは、その本来のセンサによって検出されるリンクBをセンサとして定義し、そのリンクBの姿勢に応じて、図13（A）、図14（A）の場合はオフ、図13（B）、図14（B）の場合はオンと定義される。このように、センサの場合も、モータの場合と同様、本来のセンサではなくそのセンサにより検出されるリンクをセンサとして定義し、そのセンサリンクの姿勢に応じてオン、オフを定義することにより、シミュレーションが容易となる。

【0120】

図15は、モータの種類を示す模式図である。

【0121】

本実施形態では、モータを定義するにあたり「無段变速」と「有段变速」との二種類の定義方法が用意されている。

【0122】

無段变速は、速度指令値と目標速度との間の比例定数を定義する定義方法である。この無段变速はさらに、正負に関係なく比例させる方法（例えば比例定数が100のとき、二進数101（10進数で-3）が-300 rpm（マイナス符号は逆転方向への回転をあらわす）に変換される）と、二進数の最上位ビットを符号とみなす方法（例えば二進数で101のとき、最上位ビット‘1’とその他のビット‘01’とに分け、最上位ビット‘1’はマイナス符号（‘0’はプラス符号）、残りの‘01’を数値（10進数で1）とし、それらを合わせた-1に比例定数100を掛けて-100 rpmに変換する方法）との2種類が用意されている。

【0123】

有段变速は、速度指令値と目標速度との対応表が定義される。例えば図15では、101に対し-200 rpmが定義されており、速度指令値として‘101’が入力されると目標速度として-200 rpmが出力される。

【0124】

本実施形態ではこのような複数種類の定義方法を用意しておくことにより、モータの定義の自由度を向上させている。

【0125】

図16は、モータを一次遅れ系とみなしたときの、モータの回転速度ωの初期変化を示す図である。

【0126】

ここでは、モータの回転速度変化を図16に示すような一次遅れ系としてとらえて、モータの負荷による遅れやその影響を取り入れたシミュレーションを行なう。モータを一次遅れ系とみなすと、ユーザは、整定時間4Tを入力するだけでそのモータの特性を指定することができ、入力作業の繁雑さが低減する。

【0127】

以下に、モータを一次遅れ系とみなしたときの、シミュレーション間隔Δtの間のモータの回転量Δθの算出式を導出する。

【0128】

時刻tにおけるモータ速度ωは、

【0129】

【数1】

$$\omega = \omega_0 + (\omega_i - \omega_0)(1 - e^{-\frac{t}{T}}) \quad \dots \dots (1)$$

【0130】

によりあらわされる。ここで ω_i は目標速度、 ω_0 は目標速度指令前のモータ速度、Tはモータの性質を表す、整定時間4Tに比例した定数である。式(1)より、時刻tにおけるモータ速度を ω_{k-1} とすると、

【0131】

【数2】

$$\omega_{k-1} = \omega_o + (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}}) \quad \dots \dots (2)$$

【0 1 3 2】

同様に t より Δt 後の時刻 $t + \Delta t$ のモータ速度 ω_k は

【0 1 3 3】

【数 3】

$$\omega_k = \omega_o + (\omega_i - \omega_o)(1 - e^{-\frac{t+\Delta t}{T}}) \quad \dots \dots (3)$$

【0 1 3 4】

となる。

式 (3) から式 (2) を引くと

【0 1 3 5】

【数 4】

$$\begin{aligned} \omega_k - \omega_{k-1} &= (\omega_i - \omega_o)(1 - e^{-\frac{t+\Delta t}{T}}) - (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}}) \\ &= (\omega_i - \omega_o) \left\{ (1 - e^{-\frac{t+\Delta t}{T}}) - (1 - e^{-\frac{1}{T}}) \right\} \\ &= (\omega_i - \omega_o) \left(e^{-\frac{1}{T}} - e^{-\frac{t+\Delta t}{T}} \right) \\ &= (\omega_i - \omega_o) e^{-\frac{1}{T}} \left(1 - e^{-\frac{\Delta t}{T}} \right) \quad \dots \dots (4) \end{aligned}$$

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_o) e^{-\frac{1}{T}} \left(1 - e^{-\frac{\Delta t}{T}} \right) \quad \dots \dots (5)$$

【0 1 3 6】

となる。

さて、ここで (2) 式を以下のように書き換える。

【0 1 3 7】

【数 5】

$$\begin{aligned}
 \omega_{k-1} - \omega_o &= (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}}) \\
 &= (\omega_i - \omega_o) - (\omega_i - \omega_o)e^{-\frac{1}{T}} \\
 (\omega_i - \omega_o)e^{-\frac{1}{T}} &= \omega_i - \omega_o - \omega_{k-1} + \omega_o \\
 &= \omega_i - \omega_{k-1} \quad \cdots \cdots (6)
 \end{aligned}$$

【0138】

式(6)を(5)に代入することにより以下の式が得られる。

【0139】

【数6】

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\frac{\Delta t}{T}}) \quad \cdots \cdots (7)$$

【0140】

Δt の間にモータが回転する量は、式(7)を Δt について積分すれば良いから

【0141】

【数7】

$$\begin{aligned}
 \Delta\theta &= \int_0^{\Delta t} \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\frac{1}{T}}) dt \\
 &= \omega_{k-1}\Delta t + (\omega_i - \omega_{k-1})\{\Delta t - T(1 - e^{-\frac{\Delta t}{T}})\} \quad \cdots \cdots (8)
 \end{aligned}$$

【0142】

となる。モータを一次遅れ系とみなしたとき、この(8)式に基づいて、目標速度 ω_i 、シミュレーション時間 Δt 、そのシミュレーション時間 Δt だけ前の時刻におけるモータ速度 ω_{k-1} 、および整定時間 $4T$ によって、そのシミュレーション時間 Δt の間のモータの回転量 $\Delta\theta$ が求められる。

【0143】

ただし、モータは、常に上記(8)式のように動作する必要はなく、例えばモータに指令される目標速度 ω_i が変化してから整定時間 $4T$ が経過した時点で速度を ω_i に固定してもよい。

【0144】

図17は、シミュレータにおける、一次遅れを考慮したモータ駆動の処理を示すフローチャートである。

【0145】

部品間距離よりシミュレーション間隔 Δt が決定され(ステップS501)、その後さらに制御プログラムからアクチュエータ信号が入力されると(ステップS502)、図15に示すモータの定義に従って比例定数あるいは対応表が参照されて目標速度 ω_i が決定される(ステップS503)。

【0146】

ステップS504では、目標速度 ω_i が変更されたか否かが判定され、今回目標速度 ω_i が変更されていたときはステップS505に進み、目標速度 ω_i が変更された時点からの経過時間を示す t_{total} が $t_{total}=0$ に初期化され、次回のステップS504での判定のために ω_{iold} に ω_i が格納される(ステップS506)。

【0147】

ステップS504で $\omega_i = \omega_{iold}$ であると判定された場合は直接に、 $\omega_i \neq \omega_{iold}$ であったときはステップS505、S506を経由した後に、ステップS507に進み、目標速度 ω_i が変更された時点からの経過時間 t_{total} が整定時間4Tに達したか否かが判定される。 t_{total} が未だ4Tに達していないときは、ステップS508に進み、上述の式(7)に従ってモータの回転速度 ω が求められ、さらにステップS509において、式(8)に従ってモータの回転量 $\Delta \theta$ が求められる。

【0148】

一方、ステップS507において $t_{total} \geq 4T$ であると判定されたときはステップS510に進み、モータの回転速度 ω が目標速度 ω_i と等しい速度に設定され、さらにステップS511において、モータの回転量 $\Delta \theta$ が、目標速度 ω_i で時間 Δt だけ回転したときの回転量に設定される。

【0149】

ステップS508、S509を経由した場合であっても、あるいはステップS510、S511を経由した場合であっても、ステップS512において t_{total} が Δt だけ増分され、モータのトータルの回転量 θ が $\Delta \theta$ だけ増分される。

【0150】

この図17に示す処理が制御プログラムと同期をとりながら繰り返し実行されることにより、図16に示す一次遅れ系とみなしたときの速度変化に従ってモータが回転する。

【0151】

図18～図20は、干渉チェックを用いたセンサの定義方法の説明図である。

【0152】

図18には、接触センサの姿勢が複数のモータの姿勢の組合せにより決定される例が示されており、この場合、図13、図14を参照して説明した、実際のセンサにより検出されるリンクをセンサとみなす方法を採用すると、複数のモータそれぞれの姿勢の組合せでセンサのオン、オフを定義することになり極めて複雑な定義が必要となる。そこでここでは、センサと土台との干渉をチェックし、それらの間の干渉の有無によってセンサをオンあるいはオフとする。こうすることにより、ユーザに複雑な定義を要求することなく、センサのオン、オフを決定することができる。

【0153】

図19は、スリットの移動により光量センサのオン、オフが多数回繰り返される例が示されており、この場合、図20に示すように光束を1つの仮想リンクと見なし、その仮想リンクとスリットとの干渉の有無によってセンサのオン、オフを判定する。

【0154】

この場合、仮想リンクとの干渉をチェックする相手のリンクは、図19に示すように相手のリンクがあらかじめわかっているときはそのリンクに限定してもよく、あるいは相手のリンクが不明である場合であっても、本来のセンサ自身をあらわすリンクは、その仮想リンクと常に干渉(接触)しているため、干渉をチエ

ックする相手のリンクからは除外される。

【0155】

この仮想リンクによるセンサ出力決定方法は、ここに示した光電センサなどに限らず、例えばマイクロスイッチの接触部分を仮想リンクとして定義するなど、接触式の位置検出センサにも適用することができる。

【0156】

図13、図14を参照して説明したセンサの定義方法は、ポテンションメータ、エンコーダ等の角度を検出するセンサなど、一自由度の変位を検出するセンサを表現するのに有効な方法であるが、ここに示したような、複数のモータの姿勢の組合せにより出力が決定されるように配置されたセンサ（図18）など多自由度の組合せの結果として姿勢が決定されるセンサや、スリットカウンタ（図19）などオン、オフの変化が多いセンサにはその定義方法を採用するとユーザに複雑な定義を要求することになるため、ここに説明した、リンクどうしの干渉の有無によりセンサオン、オフを判定する方法を採用することが好ましい。

【0157】

図21は、シミュレータと制御プログラムとの間で同期をとるために送受信される信号のタイミングチャート、図22は、その同期を実現するための3Dモデルの動作との同期をとるための同期化プログラムを示すフローチャート、図23は、その同期を実現するための、シミュレータ側の処理を示すフローチャートである。ここで、同期化プログラムは、3Dモデルに対応する製品を製造したときのその製品の動作を制御する制御プログラムに追加されるプログラムである。

【0158】

図21に示す、ループフラグ（Loop Flag）、レディフラグ（Ready Flag）、およびタイムスケールは、それぞれ以下の意味を持つ信号である。

【0159】

(1) Loop Flag

制御プログラムからシミュレータへ受け渡す信号であって、制御プログラムが現在動作中であるか否かを表わしている。

【0160】

(2) Ready Flag

シミュレータから制御プログラムへ受け渡す信号であって、シミュレータによる1つのシミュレータ間隔 Δt 分のシミュレーションが終了したことを表わしている。

【0161】

(3) タイムスケール

シミュレータから制御プログラムへ受け渡す信号であって、シミュレータが部品間最短距離に基づいて決定したシミュレーション間隔 Δt をあらわしている。

【0162】

一般に、制御プログラムは、メインループやタイマ割り込みルーチンなど、一定周期で動作する部分が存在する。同期処理のために制御プログラムに追加される同期化プログラムは最小限の規模のものであることが好ましく、ここに示す例では、タイムスケールとして、5 msec, 2 secなどの具体的な時間ではなく、この一定周期で動作するルーチンを通過する回数nで指定している。

【0163】

図22に示すルーチンはそれをあらわしており、ステップS602でn=0か否かを判定し、n=0でないときはステップS603に進んでnを1だけデクリメントし、本来の制御プログラムの処理の実行に移っている。ステップS601においてn=0と判定されたときのみ、ステップS603～S607の同期のための処理が実行されるように構成されている。

【0164】

ここでは、図21～図23を説明するにあたり、シミュレータでは現在シミュ

レーションが行なわれており、したがって **Ready Flag** は ‘Low’ レベルにあり、制御プログラム側では図22のステップS604にとどまって **Ready Flag** が ‘High’ に変化するを待っている段階にあるとする。このとき、シミュレータにおける同期処理をあらわす図23のルーチンでは、ステップS703のシミュレーションを行なっている途中にある。

【0165】

ここで、シミュレーションが終了すると、シミュレータは図23のステップS704に移り、次のシミュレーション間隔（タイムスケール） Δt （ただし上述したように制御プログラムの動作回数nであらわされている）が決定され、ステップS705においてそのタイムスケール Δt （n）が制御プログラムに送信され、さらにステップS706に進んで **Ready Flag** が ‘High’ レベルに変更される。

【0166】

すると、制御プログラム側では、図22に示すステップS604を抜け出してステップS605に進みタイムスケールnが読み込まれ、さらにステップS606に進んで **Loop Flag** が ‘High’ に変更される。その後ステップS607において、シミュレータ側から送信されてきている **Ready Flag** が ‘Low’ に変化するのを待って本来の制御処理に進む。前述したように、この制御プログラムは一定周期で繰り返し実行されており、ステップS601では $n=0$ か否かが判定され、 $n \neq 0$ のときはステップS602に進んでnが1だけデクリメントされる。これを繰り返し、ステップS601で $n=0$ と判定されるとステップS603に進み、 **Loop Flag** が ‘Low’ に変更され、その後、ステップS604において、シミュレータ側から送られてくる **Ready Flag** が ‘High’ に変化するのを待つ状態となる。

【0167】

一方、シミュレータ側では、図23のステップS706において **Ready Flag** が ‘High’ に変更された後、ステップS707において、制御プログラム側から送られてくる **Loop Flag** が ‘High’ に変化するのを待ち、 **Loop Flag** が ‘High’ に変化したことを見てステップS708に進み、 **Ready Flag** を ‘Low’ に変化させ、ステップS702に進んで、 **Loop Flag** が ‘Low’ に変化するのを待つ。 **Loop Flag** が ‘Low’ に変化すると、ステップS703に進み、シミュレーション間隔 Δt 分のシミュレーションが行なわれる。

【0168】

以上の処理が繰り返され、シミュレータと制御プログラムでは、同期がとられながらそれぞれの処理が実行される。

【0169】

次に、リンク（部品）どうしの干渉可能性の評価方法について説明する。

【0170】

図24は、干渉可能性の評価方法の説明図である。ここには、リンクAが速度vでリンクBに向かって動いている例が示されている。

【0171】

図24（A）は干渉する可能性のある2つのリンク（リンクAとリンクB）が大きく離れている状態を示しており、この場合は干渉可能性が小さいものと判定されてシミュレーション間隔 Δt とし比較的大きな値が設定され、リンクAは一回のシミュレーションで $v \Delta t$ だけ進む。この場合、シミュレーションを高速に行なうことができる。

【0172】

また、図24（B）は、リンクAとリンクBとが近づき、それらの間の距離が小さい状態をあらわしている。この場合、干渉可能性が大きいものと判定され、シミュレーション間隔 Δt として比較的小さな値が設定される。このときは、リンクAは、 Δt が小さいため一回のシミュレーションで進む距離 $v \Delta t$ が小さく

、高精度なシミュレーションを行なうことができる。

【0173】

このように、本実施形態では、部品間の距離に応じてシミュレーション間隔 Δt が変更され、高精度なシミュレーションと高速なシミュレーションとの両立、調和が図られている。

図25は、部品間の距離dとシミュレーション間隔 Δt との関係の各種の例を示す図である。

【0174】

これらの例に示すように、距離dが小さくてもシミュレーションがあまりにも低速化されないよう、かつ距離dが大きくてもシミュレーションがあまりにも高速化されないようにその調整が図られている。

【0175】

次に、干渉可能性の評価のために部品をグループ分けする方法について説明する。

【0176】

図26は、部品のグループ分けの説明図である。

【0177】

ここにはリンク 1_0 ～リンク 1_5 の5つのリンクが示されており、モータリンクとして定義されたリンク 1_3 が動く場合を考える。この場合、モータリンク 1_3 の動きによって姿勢に影響を受けるリンクはリンク 1_3 ～ 1_5 であり、これらのリンク 1_3 ～ 1_5 は、モータリンク 1_3 の動きに対して1つの剛体と考えることができる。また、モータリンク 1_3 の動きによっては姿勢に影響を受けないリンク 1_0 ～ 1_2 も同様に、これを合わせたものを1つの剛体と考えることができる。そこでモータリンク 1_3 を動作させる場合には、そのモータリンク 1_3 を動かしたことにより姿勢に影響を受けるリンクのグループと影響を受けないリンクのグループとの2つのグループに分けてそれら2つのグループそれぞれを剛体とみなし、それら2つの剛体の間の最短距離を求めるこにより、その求めた距離を次のシミュレーション間隔 Δt を決定するための最短距離の候補の1つとすることができる。ただし、この図26におけるリンク 1_3 とリンク 1_2 、例えばモータの軸と歯車等は、常に接触しているにも拘らず異なるグループに分けられてしまい、このままでは上記2つの剛体は常に干渉していることになってしまふ。そこでここでは、上記のようにして2つのグループに分けた後、一度部品間距離を求め、既に接触、干渉しているリンクのうち、今注目しているモータリンクを動かしたときに動かない方のグループに属するリンク（図26に示す例の場合リンク 1_2 ）をそのグループから除外した上で2つのグループそれぞれを剛体とみなし、それら2つの剛体の間の距離が求められる。

【0178】

図27は、1つの3Dモデル内に複数のモータリンクが定義されている場合のグループ分けの説明図である。

【0179】

図27（A）は、3Dモデルをあらわしており、ここには、モータ 1 ～モータ 3 の3つのモータリンクが定義されている。このとき、図26を参照して説明したグループ分けを、モータ 1 のみを動かしたとき（図27（B））、モータ 2 のみを動かしたとき（図27（C））、モータ 3 のみを動かしたとき（図27（D））のそれぞれについて行なう。こうしておいて、シミュレーション実行時には、対象としているモータリンクを動かしたときは、そのモータリンクに関しグループ分けしたグループ間の最短距離が求められ、それら複数のモータリンクの動きにより求められた複数の‘最短距離’の中からさらに最短の距離が求められる（図10ステップS201～S207参照）。

【0180】

図28は、グループ分けした部品の再グループ化の説明図である。

【0181】

図27を参照して説明した手法によると、各モータリンクに関し、その3Dモデルを構成するほぼ全ての部品がどちらかのグループ(剛体)に属する結果となり、この場合、各剛体を定義するためのデータ量が膨大なものとなってしまい極めて大きなメモリ容量を必要とする可能性がある。そこで、図28に示すように、各モータリンクの動きによって動かない方のグループ(図28に示す例ではグループD, E, F)に属する部品群のうち、共通部分を抜き出して1つの剛体とともに、各グループD, E, Fからその共通部分を除いた部分をそれぞれ1つの剛体とする。こうしておいて、距離の測定の際は、共通部分からなる剛体とその共通部分を除いた部分の剛体とを合体させて、相手のグループ(グループD, E, F)に対応してそれぞれグループA, B, Cとの間の距離を求めるようにもよい。こうすることにより、共通部分を各グループに配置しておく場合(図27の場合)と比べ、これらのグループを定義するデータを格納しておくためのメモリの容量を減らすことができる。

図29は、モータリンクの動きによって姿勢に影響を受けるリンクのグループとそのモータリンクの動きによっては姿勢に影響を受けないグループとに分けるルーチンを示すフローチャート、図30は、そのサブルーチンである検索ルーチンのフローチャートである。

【0182】

ここでは、モータリンクの動きによって姿勢に影響を受けるリンクがリストAにリストアップされ、それ以外のリンクがリストBにリストアップされる。尚、リストAおよびリストBは、ここで説明するフローの実行に先立って白紙状態にクリアされているものとする。

【0183】

ここでは、先ずリンクを指し示すポインタPによりモータリンクが指し示され(図29、ステップS801)、そのポインタPで指し示されるリンクがリストAに追加される。

【0184】

次にポインタPで指し示されているリンクにチャイルド(child)のリンクが存在しているか否かが判定される。childとは、例えば図26に示すリンク機構においてポインタがモータリンク₁₃を指し示していたとき、そのモータリンク₁₃が動くことによって一緒に動くリンクの1つ(ここではリンク₁₄)をいう。リンク₁₄が仮にモータリンクであったとしても今着目しているモータリンクはリンク₁₃であり、リンク₁₄はリンク₁₃のchildとなる。尚、ポインタPが指し示すリンクに同格の複数のchildが考えられるときであってもchildはそのうちの1つに限定され、childと考えられる他のリンクは、その1つの限定されたchildのブラザ(brother)として定義される。

【0185】

ポインタPが指し示すリンクにchilldがあったときは、ステップS804に進み、そのchilldのリンクを指し示すようにポインタPを移し、図30に示す検索ルーチンが呼ばれる。検索ルーチンについては後述する。検索ルーチンの実行が終了してステップS805に戻ってくると次にステップS806に進む。尚、ステップS803においてポインタPが指し示すリンクにchilldが存在していないときは直接にステップS806に進む。

【0186】

ステップS806では、ポインタPで指し示されたリンクがギアやカム等の受動部品を駆動する駆動リンクであるか否かが判定される。

【0187】

ポインタPが受動部品を駆動する駆動リンクであったときは、ステップS807に進み、ポインタPが指し示すリンクによって駆動される受動部品であるリンクを指し示すようにポインタPが変更され、ステップS808において図30の検索ルーチンが呼ばれる。検索ルーチンの実行が終了してステップS808に戻

ってくると次にステップS 809に進む。尚、ステップS 806において、ポインタPが指示するリンクが受動部品を駆動するリンクではないと判定されると、直接にステップS 809に進む。

【0188】

このステップS 809にまで進んだ段階では、リストAが完成しており、このステップS 809では、全リンクからリストAにリストアップされたリンクを除いたリンクがリストBに格納される。次いで、リストAにリストアップされた全てのリンクとリストBにリストアップされた全てのリンクとの間で干渉チェックが行なわれ（ステップS 810）、干渉していたリンクの組のうち、リストBにリストアップされた方のリンクがリストBから削除される。

【0189】

図30に示す検索ルーチンでは、先ずステップS 901において、ポインタPが指示するリンクがリストAに既にリストアップされているか否かが判定され、既にリストアップされていたときはそのままこの検索ルーチンを抜けてこの検索ルーチンが呼ばれた元のルーチンに戻る。ステップS 901において、ポインタPが指示するリンクがリストAにリストアップされていなかったときはステップS 902に進み、ポインタPが指示するリンクがリストAに追加される。ステップS 903ではポインタPが指示するリンクにchildが存在するか否かが判定され、childが存在していたときにはステップS 904に進んでポインタPがそのchildのリンクを指示するように変更され、ステップS 905において再度この図30に示す検索ルーチンが呼ばれる。このように、この検索ルーチンは何重にも重なって呼ばれることがあり、この検索ルーチンの実行が終了すると、その呼ばれた先（例えばステップS 905で呼ばれたときはそのステップS 905）に戻る。

【0190】

ステップS 906では、ポインタPが指示するリンクにbrotherが存在するか否かが判定され、brotherが存在していたときは、ステップS 907に進んでポインタPがそのbrotherのリンクを指示するように変更され、ステップS 908において再度この検索ルーチンが呼ばれる。

【0191】

ステップS 909では、ポインタPで指示されるリンクが、ギアやカム等の受動部品を駆動する駆動リンクであるか否かが判定され、駆動リンクであったときはステップS 910に進んでポインタPがその駆動リンクを指示するように変更され、ステップS 911においてこの検索ルーチンが再度呼ばれる。

【0192】

図31は、図29、図30に示すルーチンの動作説明用のリンク機構モデルを示す図、図32は、図31に示すリンク機構モデルのデータ構造を示す図である。

【0193】

図31に示すように、リンクAはモータ、リンクBおよびリンクCは、それぞれ、リンクA（モータ）の回転に伴って一緒に回転するシャフトおよびギア、リンクDはリンクCと噛合しているギアである。リンクEはリンクD（ギア）の回転に伴って一緒に回転するシャフト、リンクFはリンクD（ギア）と噛合しているギアである。

【0194】

データ構造上は、図32に示すように、リンクAとリンクDとリンクFが同格に並び、リンクBはリンクAのchild、リンクCはリンクBのbrother、リンクEはリンクDのchildの関係にある。

【0195】

図31、図32に示すリンク機構モデルについて、図29、図30に示すルーチンの実行によりリストAを作成することを考える。ここではリンクA（モータ）から検索が始められる。すなわち、図29のステップS 801においてリンクA

が指し示されるようにポインタ P が指定される。以下のかっこ内は、リスト A にリストアップされるリンクをあらわしている。

【0196】

- (1) P = A でスタート (A)
- (2) P が A からその child B に変更され、検索ルーチンが実行される (A, B)
- (3) P が B から brother C に変更され、検索ルーチンが実行される (A, B, C)
- (4) P が C の受動部品である D に変更され、検索ルーチンが実行される (A, B, C, D)
- (5) P が D からその child E に変更され、検索ルーチンが実行される (A, B, C, D, E)
- (6) return で P が D に戻る。

【0197】

- (7) P が D から D の受動部品である F に変更され、検索ルーチンが実行される (A, B, C, D, E, F)
- (8) return で P が D に戻る。
- (9) return で P が C に戻る。
- (10) return で P が B に戻る。
- (11) return で P が A に戻る。
- (12) 終了

以上の手順を踏んで、図31に示す全てのリンク（リンク A～リンク F）がリスト A にリストアップされる。

【0198】

図33は、1つのモータリンクの動きによって姿勢の変化を受けるグループ内の干渉可能性の評価方法の説明図である。

【0199】

モータリンクの動きによって干渉する可能性があるのは、図26を参照して説明したような、そのモータリンクの動きによって姿勢に影響を受けるリンクのグループと姿勢に影響を受けないリンクのグループとの間のみでなく、図31に示すように姿勢に影響を受けるグループの内部でも干渉が生じる可能性がある。そこで、そのグループの内部で干渉する可能性のあるリンクの組を発見するために、ここでは、シミュレーションに先立って、モータリンクを少しずつ動かしながら、そのモータリンクを動かすことによって姿勢に影響を受けるグループ内の全てのリンクどうしの間の干渉チェックを行う。このような総あたり的な干渉チェックによって干渉したリンクの組は実際のシミュレーションにあたっても相互間の距離を求める対象に加え、シミュレーション間隔 Δt を決定するための最短距離の候補の1つとする。こうすることによって正確なシミュレーションを行なうことができる。ただし、モータリンクを動かしたときに姿勢に影響を受けるグループ内において、例えば図31に示すモータと歯車のように、常に接触しているリンクの組も存在する。そこでモータリンクを少しずつ動かして干渉チェックを行った際そのモータリンクの全ての姿勢で常に干渉・接触していたリンクの組、例えば歯車等で関係づけられたリンクの組はリンク間の距離を求める対象から排除される。

【0200】

図34は、モータリンクの動きによって姿勢に影響を受けるリンクグループ内で、シミュレーション時に距離を求めるリンクの組を抽出する手法の説明図である。

【0201】

図34 (A) はある1つのモータの動きによって姿勢に変化を受けるグループを構成する全部品を示しており、ここでは一例として、図34 (B) に示すように、そのモータを -180° から 180° まで 36° おきに回転させ、各回転ご

とに干渉チェックを行なうこととする。この場合、モータの回転、リンク内の干渉チェックは11回行なわれることになる。すると、図34 (C) に示すような、干渉したリンクの組と干渉回数のデータが得られる。干渉チェックは11回行なわれており、リンクBとリンクE、およびリンクDとリンクEは11回干渉が発生していたため、図34 (D) に示すように、これら2つの組はシミュレーション時における距離測定の対象からは除外される。

【0202】

次に本発明の機構制御プログラム開発支援システムを構成する三次元機構モデルシミュレータの別の実施形態について説明する。

【0203】

図35は、本実施形態におけるシミュレーション全体の処理の流れを示す図である。

【0204】

ここでは、制御プログラムとの同期処理（ステップS1001）、モータ処理（ステップS1002）、関節不良動作処理（ステップS1003）、センサ処理（ステップS1004）、および制御プログラムの実行時間△Tの決定処理（ステップS1005）が繰り返し実行される。

【0205】

制御プログラムとの同期処理（ステップS1001）に関しては、前述の実施形態と同一であるため（例えば図9、図21を参照）、ここでは説明は割愛する。それ以外の処理に関しては、以下順次説明する。

【0206】

図36は、モータ処理ルーチンを表わすフローチャートである。このフローチャートは、基本的には、前述した実施形態における、図10の一点鎖線で囲った部分からなるモータ処理にモータの故障の概念を取り入れたものである。

【0207】

ステップS1101では、最短距離Dを求めるために先ずそのDに∞が初期入力され、ステップS1102では、未処理のモータが存在しているか否かが判定され、未処理のモータ1つずつについてステップS1103～S1110の処理が行なわれる。

【0208】

即ち、先ず未処理のモータそれぞれについて、そのモータが故障しているか否かが判定される（ステップS1003）。モータの故障の有無は、ユーザによりあらかじめ設定され、フラグとして保存されている。故障のモータについては処理は行なわれず、故障でないときは、制御プログラム側から送られてきた、今処理しようとしているモータに関するアクチュエータ信号に応じて、そのモータの姿勢変位量△θが決定され（ステップS1004）、△θ=0でなければ（ステップS1105）、そのモータのそれまでの姿勢θに今回の変位量△θが加算され（ステップS1106）、そのモータリンクの移動に伴って移動するリンクの姿勢変位量を計算するリンク姿勢移動サブルーチン（図37参照、後述する）が実行され（ステップS1107）、部品間最短距離dが求められ（ステップS1108）、d < Dか否かが判定され、d < Dのとき、すなわち、今回求めた最短距離dがこれまでに求めた最短距離Dよりも短かいときは（ステップS1109）、Dにdを格納して最短距離Dを更新する（ステップS1110）。

【0209】

全てのモータに関し、今回の演算時間△Tの間の姿勢変更が終了すると、このモータ処理を終了する。

【0210】

図37は、図36に示すモータ処理ルーチンのステップS1107で実行されるリンク姿勢移動サブルーチンのフローチャートである。

【0211】

先ず、リンク姿勢移動（図37のステップS1107でコールされたときは、

モータリンクの姿勢移動) を受けて (ステップS1201) 、その姿勢移動のあったリンクに対する受動リンクに未計算の受動リンクが存在するか否か判定し (ステップS1202) 、未計算の受動リンクが存在するときは、今回姿勢移動のあったリンクとそのリンクに対する受動リンクとの間にクラッチが存在するか否かを判定し (ステップS1203) 、クラッチが存在するときは、そのクラッチ部品の姿勢がオンの位置にあるか否かを判定する (ステップS1204) 。クラッチ部品の存在、およびクラッチ部品の姿勢とクラッチのオン、オフとの関係は、ユーザによりあらかじめ定義される。

【0212】

クラッチ部品が存在しないとき、あるいはクラッチ部品が存在していてもオン状態にあるときは、ステップS1205に進み、ギア、カム関係より受動部品の移動量を計算し、さらにステップS1206において、今回移動量を計算した受動部品を能動部品としたときのその能動部品に対する受動部品の移動量を求めるために、再度このリンク姿勢移動サブルーチンがコールされる。

【0213】

このようにして、このリンク姿勢移動サブルーチンは、図30に示す検索ルーチンと同様、再帰的に実行され、この再帰的な実行により、1つのモータを動かしたとき他の部品の移動量が高速に求められる。

【0214】

図38は、図35のステップS1003で実行される関節不良動作処理ルーチンのフローチャートである。

【0215】

ここでは、先ず、三次元機構モデルが画面表示され (ステップS1301) 、その後、ユーザが新たに不良リンクを指定したか否か (ステップS1302) 、未処理のユーザ指定不良リンクが存在するか否か (ステップS1305) 、未処理の、干渉が起きたリンクが存在するか否か (ステップS1310) 、および不良リンクが存在するか否か (ステップS1314) が判定される。

【0216】

ステップS1302において、ユーザが新たに不良リンクを指定したと判定されると、ステップS1303に進み、その新たに指定された不良リンクを駆動しているモータを検索し、そのモータとそのモータの移動方向をリストに追加する (ステップS1304) 。

【0217】

ステップS1305において未処理のユーザ指定不良リンクが存在すると判定されると、ステップS1306に進み、その不良リンクを駆動しているモータの移動方向が逆転したか否かが判定され、逆転していないときはステップS1307に進んでそのモータの姿勢を旧姿勢に戻し、そのモータの姿勢が旧姿勢に戻されたことに対応してリンクの姿勢を旧姿勢に戻すために、図37に示すリンク姿勢移動サブルーチンがコールされる (ステップS1308) 。一方、ステップS1306においてそのモータの移動方向が逆転したと判定されると、ステップS1309に進み、そのモータを不良リンクのリストから削除する。

【0218】

こうすることにより、不良リンクがある一方向に移動しようとしたときに何らかの原因で引っ掛けあって動かなくなり、その不良リンクが一旦逆方向に移動することによりその引っ掛けかりが解除されて正常に戻るという動作がシミュレートされる。

【0219】

ステップS1310において、未処理の、干渉が起きたリンクが存在する旨判定されると、ステップS1311に進み、その不良リンクを駆動しているモータの検索が行われ、そのモータの姿勢が旧姿勢に戻され (ステップS1312) 、そのモータによって駆動される他のリンクの姿勢を旧姿勢に戻すべく、図37に示すリンク機構サブルーチンがコールされる (ステップS1313) 。

【0220】

これにより、他のリンクと干渉してそれ以上動かすことのできない状態をシミュレートすることができる。

【0221】

ステップS1314において不良リンクが存在すると判定されると、その三次元機構モデルを画像上に再描画する（ステップS1315）。こうすることにより、ステップS1301における描画と合わせ、シミュレーション画面上で見ると不良リンクを受動部品に持ったモータおよびそのモータにより駆動される全リンクの動きがカクカクし、いかにもひつかかった様子が表現される。

【0222】

尚、ここに示した例では、モータの姿勢を元に戻すことによって関節不良を実現したが、関節値の上限あるいは下限を一時的に現在値に設定することで関節不良を実現することも可能である。

【0223】

図39は、図35のステップS1004で実行されるセンサ処理ルーチンのフローチャートである。前述の実施形態では、センサはオン、オフの信号を出力するオン／オフセンサに限られていたが、ここでは、センサとして、オン／オフセンサのほか、ポテンショメータおよびエンコーダも存在し、さらにセンサの不良も考慮されている。

【0224】

ステップS1401では、未処理のセンサの有無が判定され、未処理のセンサが存在すると、ステップS1402に進み、その未処理のセンサがオン／オフセンサであるか否かが判定される。オン／オフセンサであったときはステップS1403において、オン／オフセンサ処理が行なわれ、そのオン／オフセンサを処理した結果のセンサ値（オンあるいはオフ）が出力される。

【0225】

ステップS1402において、その未処理のセンサがオン／オフセンサではないと判定されると、ステップS1404に進み、そのセンサがポテンショメータであるかエンコーダであるかが判定され、ポテンショメータであると判定されるとステップS1405に進んでポテンショメータ前処理が行なわれた後、エンコーダであると判定されると直接に、ステップS1406のエンコーダ処理に進み、その結果としてのセンサ値が出力される。

【0226】

図40は、図39のステップS1403で実行されるオン／オフセンサ処理ルーチンのフローチャートである。

【0227】

このオン／オフセンサ処理ルーチンではそのオン／オフセンサの故障もシミュレートされており、先ずステップS1501、1503でそのオン／オフセンサがオンの状態のまま故障しているあるいはオフの状態のまま故障しているかが判定され、オン状態のまま故障しているときは出力予定値としてオンが設定され（ステップS1502）、オフ状態のまま故障しているときは出力予定値としてオフが設定され（ステップS1504）、ステップS1508に進んでその出力予定値がセンサ値として出力される。そのオン／オフセンサがオン状態あるいはオフ状態で故障しているか否かはユーザによりあらかじめ設定される。

【0228】

そのオン／オフセンサが故障していないときはステップS1505に進み、そのセンサリンクの姿勢から出力予定値（オンあるいはオフ）が決定され（ステップS1505）、次いで性能劣化による出力予定値変更処理（ステップS1506）、およびチャッタリングによる出力予定値変更処理（ステップS1507）が実行される。

【0229】

図41は、図40のオン／オフセンサ処理ルーチンのステップS1506で実

行されるセンサ性能劣化処理ルーチンのフローチャートである。このルーチンでは、オン／オフセンサが故障（出力がオンまたはオフに固定）までいかなくとも、応答性能が劣化し、そのオン／オフセンサの出力をオンまたはオフに変化させる状態が生じた後、暫らくは前の状態を保ち、遅れてオンまたはオフに変化する状態がシミュレートされる。

【0230】

ステップS1601では、旧出力予定値dと出力予定値とが等しいか否かが判定され、変化していたときはステップS1602に進み出力予定値がオンかオフかが判定される。出力予定値がオンのときはステップS1603に進み、センサ出力がオフからオンに変化するときの遅れ時間がTdに設定され、一方出力予定値がオフのときはステップS1604に進み、センサ出力がオフからオンに変化するときの遅れ時間がTdに設定される。

【0231】

ステップS1605ではTd>0か否か判定され、Td>0のときは、ステップS1606に進んでTdから制御プログラムの実行時間ΔTが減算され、ステップS1607において出力予定値が反転される（出力予定値がオンのときはオフ、出力予定値がオフのときはオン）。

【0232】

ステップS1608では、旧出力予定値dに今回の出力予定値が格納される。

【0233】

図35に示すように、センサ処理S1004には繰り返し実行され、したがって図41に示すセンサ性能劣化処理も繰り返し実行され、Td>0のときは、この処理が繰り返されるたびにTdからΔTが減算されTd≤0に達するとセンサが変化することになる。

【0234】

図42は、図40のオン／オフセンサ処理ルーチンのステップS1507で実行される、チャッタリングによる出力予定値変更処理ルーチンのフローチャートである。このルーチンではオン／オフセンサがオンからオフ、あるいはオフからオンに変化するときのチャッタリングがシミュレートされる。

【0235】

ステップS1701では、旧出力予定値cと出力予定値とが等しいか否かが判定され、変化していたときはステップS1702に進み、出力予定値がオンかオフかが判定される。出力予定値がオンのときはステップS1703に進み、センサ出力がオフからオンに変化するときのチャッタリング時間がTcに設定され、一方出力設定値がオフのときはステップS1704に進み、センサ出力がオンからオフに変化するときのチャッタリング時間がTcに設定される。

【0236】

ステップS1705では、旧出力予定値cに出力予定値が設定される。

【0237】

ステップS1706では、Tc>0か否かが判定され、Tc>0のときはステップS1707に進んでTcから制御プログラムの実行時間ΔTが減算され、ステップS1708において0～99の値をとる乱数が発生されてその発生した乱数があらかじめ設定された発生確率（0～99のうちのいずれかの値）と比較され、乱数>発生確率のときは出力予定値のオン、オフが反転される（ステップS1709）。

【0238】

このようにして、Tc>0の間、確率的にオン、オフが変化するチャッタリングがシミュレートされる。

【0239】

次に、図39のステップS1406で実行されるエンコーダ処理について説明する。

【0240】

図43は、エンコーダの変位（角度等）に対する出力波形の一例を示す図、図44は、エンコーダの、各種の出力波形を示す図である。

【0241】

ここでは、図43に示すように、エンコーダの出力波形は、波長Lの周期的な関数 $f(\theta)$ をとるものと考える。ただし関数 $f(\theta)$ は、図44に示すように各種の形をとることができ。ここで波長Lは、関数 $f(\theta)$ の一周期をあらわし、例えば、このエンコーダがロータリ式のエンコーダであって回転角度1.0度ごとに一周期の信号を出力するエンコーダであるときは、 $L = 1.0$ (度) となる。

【0242】

ここで、変位 θ を電気角 $\phi = 2\pi\theta/L$ に変換し、関数 $f(\theta)$ を電気角 ϕ であらわしたとき $g(\phi)$ と表記するものとして、以下、エンコーダ機能の原理を説明する。

【0243】

前回行なったシミュレーション時のエンコーダの電気角を ϕ 、今回すなわち前回行なったシミュレーションから制御プログラムの演算時間 ΔT だけ進んだ時刻の電気角を ϕ 、エンコーダの電気角速度を ω とすると、前回から今回までの間のシミュレーションサイクルにおけるエンコーダの移動量は $\omega \Delta T$ となり、

$$\phi = \phi_0 + \omega \Delta T$$

が成立する。

【0244】

しかし、エンコーダはパルス数を測定するため、今回のシミュレーションサイクルにおける $\omega \Delta T$ が 2π より大きい場合、単純に $g(\phi)$ を出力するわけにはいかず、所定のパルス数のパルスを出力した後に $g(\phi)$ を出力する必要がある。

【0245】

図45は、エンコーダの出力処理ルーチンのプログラムを示す図である。

【0246】

ここでは先ず前回のシミュレーション終了時のエンコーダ出力 $g(\phi_0)$ を先ず出力し（ステップS1801）、今回のシミュレーション終了時の電気角 ϕ との差分 $\phi - \phi_0$ が $\phi - \phi_0 > \pi/2$ であるか（ステップS1802）、あるいは $\phi - \phi_0 > \pi/2$ であるか（ステップS1804）が判定され、 $\phi_0 - \phi > \pi/2$ のときはステップS1803に進んで、 ϕ_0 に $\pi/2$ が加算され、ステップS1806において $50\mu\text{sec}$ 待機した後、ステップS1801に戻って新たな ϕ に対する $g(\phi)$ が outputされる。 $\phi_0 - \phi > \pi/2$ のときは、ステップS1803に進み ϕ_0 から $\pi/2$ が減算され、その後は同様にステップS1806において $50\mu\text{sec}$ 待機した後、ステップS1801に戻って新たな ϕ に対する $g(\phi)$ が outputされる。

【0247】

すなわち、ここでは $|\phi - \phi_0| > \pi/2$ のときは、 $|\phi - \phi_0| \leq \pi/2$ になるまで、 5kHz の周波数を上限として（ステップS1806）、 ϕ が $\pi/2$ ずつ変化したときの離散的な周期信号が outputされる。

【0248】

$|\phi - \phi_0| \leq \pi/2$ に達すると、ステップS1807に進み、今回のシミュレーションにおけるエンコーダの最終値である $g(\phi)$ が outputされ、次回のシミュレーションのために今回のシミュレーションにおける ϕ が ϕ_0 に格納される（ステップS1808）。

【0249】

このエンコーダ信号は、制御プログラムとは独立して動作するカウンタ回路によって受信されることを想定しており、したがってエンコーダ信号発信時に制御プログラムが停止していても問題はない。

【0250】

また、図45のフローチャートは、パルス信号を $g(\phi)$ の1つのみ出力するタイプのエンコーダを表現した例であるが、 $g(\phi)$ の他に $g(\phi + \pi/2)$ を出力することにより、A相、B相の二相信号を出力するタイプのエンコーダも表現することができる。

【0251】

さらに $g(\phi)$ のほかに $\phi - \phi_0$ の正負の符号をオン/オフで出力することにより、アップ/ダウン信号を出力するタイプのエンコーダも表現することができる。

【0252】

ここで、上記のシミュレーションに先立って、パルスの波形（図44参照）や分解能や電気角 ϕ がゼロになる位置をあらわすオフセットを入力することでエンコーダの特性がユーザにより定義される。

【0253】

図46は、図39に示す処理ルーチンのステップS1406で実行されるエンコーダ処理ルーチンのフローチャートである。

【0254】

ここではエンコーダの故障についても考慮されており、エンコーダに故障があったときの三次元機構モデルのシミュレーションのために、あらかじめユーザにより以下の故障フラグが設定される。

【0255】

故障フラグ0: 正常

- 1: A相が V_{max} で故障
- 2: A相が V_{min} で故障
- 3: B相が V_{max} で故障
- 4: B相が V_{min} で故障
- 5: アップ/ダウン信号故障 (Hig hに固定)
- 6: アップ/ダウン信号故障 (Lowに固定)

図46のステップS1901では、先ず変位 θ が電気角 ϕ に変換され、次いで、故障フラグが1であるか否か（ステップS1902）、2であるか否か（ステップS1903）が判定され、それらの何れでもないときは、ステップS1904に進んで、図45に示す $g(\phi)$ 出力処理ルーチンが実行される。一方、ステップS1902において故障フラグが1であると判定されるとステップS1905に進み、 V_{max} で故障していることに対応して $g(\pi/4)$ が出力され、一方、ステップS1903において故障フラグが2であると判定されるとステップS1906に進み V_{min} で故障していることに対応して $g(3\pi/4)$ が出力される。

【0256】

ステップS1907では、このエンコーダが2相出力タイプのエンコーダであるか否か、ステップS1908では、このエンコーダがアップ/ダウン信号を出力するタイプのエンコーダであるか否かが判定され、いずれでもない、単純に $g(\phi)$ のみを出力するタイプのエンコーダのときはそのままこのルーチンを抜ける。

【0257】

ステップS1907においてこのエンコーダが2相出力タイプのエンコーダであると判定されると、ステップS1909において故障フラグが3であるか否か、ステップS1910において故障フラグが4であるか否かが判定される。それらのいずれでもなかったときは、ステップS1911に進み、図45における ϕ を $\phi + \pi/2$ に置き換えた出力処理ルーチンが実行される。ステップS1909において故障フラグが3であると判定されるとステップS1912に進み、B相が V_{max} で故障していることに対応して $g(\pi/4)$ が出力され、これと同様に、ステップS1910において故障フラグが4であると判定されるとステップS1913に進み、B相が V_{min} で故障していることに対応して $g(3\pi/4)$ が

出力される。

【0258】

また、ステップS1908においてこのエンコーダがアップ／ダウン信号出力タイプのエンコーダであると判定されると、ステップS1914に進み故障フラグが5であるか否かが判定され、ステップS1915では故障フラグが6であるか否かが判定される。それらのいずれでもないときはステップS1916に進みそのエンコーダの移動方向が判定され、アップ方向のときはステップS1917に進んで‘H i g h’が出力され、ダウン方向のときはステップS1918に進んで‘L o w’が出力される。一方、ステップS1914において故障フラグが5であると判定されるとステップS1917に進み、エンコーダの移動方向のいかんによらず‘H i g h’が出力され、これと同様に、ステップS1915において故障フラグが6であると判定されるとステップS1918に進み、エンコーダの移動方向のいかんによらず‘L o w’が出力される。

【0259】

このようにしてエンコーダを表現するとともにエンコーダの故障を表現することができる。ただし、エンコーダの場合、オン／オフセンサの場合と異なり、処理に入った直後に故障中を示すのではなく、センサ信号を出力する際に故障フラグの判定、処理が行なわれる。

【0260】

図47は、ポテンショメータの関数値の一例を示す図である。

【0261】

ここでは、ポテンショメータを、図44(d)に示す三角波を出力する、分解能が極端に低いエンコーダとして実現している。例えば図43に示す波長Lとして360度を設定すれば、 $\theta_{\max} - \theta_{\min} = 180$ 度の変位で出力電圧Vが最小値V_{min}から最大値V_{max}まで変換するポテンショメータを表現することができる。

【0262】

図48は、図39に示すセンサ処理ルーチンのステップS1405で実行されるポテンショメータ前処理ルーチンのフローチャートである。

【0263】

ポテンショメータの場合、図47に実線で示すように、 θ_{\min} 以下のθでは出力電圧Vは最低値に固定され、 θ_{\max} 以上のθでは出力電圧Vは最大値に固定されることがあり、この図48ではそのようなポテンショメータを実現している。

【0264】

すなわち、ステップS2001において $\theta > \theta_{\max}$ を満足するか否かが判定され、それを満足するときはステップS2002に進んでθが θ_{\max} に固定される。

【0265】

またこれと同様に、ステップS2003において $\theta < \theta_{\min}$ を満足するか否かが判定され $\theta < \theta_{\min}$ を満足するときはステップS2004に進んでθが θ_{\min} に固定される。

【0266】

ポテンショメータの場合、このような前処理が行なわれた後、図46に示すエンコーダ処理が実行される(図39参照)。

【0267】

このタイプのポテンショメータの場合、ユーザは、 θ_{\min} , θ_{\max} および, V_{min}, V_{max}を入力すると、システムは自動的に三角波 f (θ) の波長、振幅、オフセットに変換する。

【0268】

尚、ストップがないポテンショメータなど、出力が周期的に変化するポテンショメータの場合は、分解能が低い三角波や鋸波を出力するエンコーダとして取り扱われる。

【0269】

図49は、図35のステップS1005において実行される ΔT 決定処理ルーチンのフローチャートである。

【0270】

ここでは、先ずステップS2101において、性能劣化（応答遅れ、あるいはチャッタリング）を起こしたセンサが存在するか否かが判定され、そのようなセンサが存在しないときは、図36に示す処理で求めた部品間の最短距離Dに応じた ΔT が決定され（ステップS2102）、一方、性能劣化を起こしたセンサが存在するときは、 ΔT としてとり得る値の最小値が設定される。

【0271】

ここで、性能劣化を起こしたセンサが存在するときに ΔT を最小値に設定するのは、性能劣化の表現精度を高め、正確なシミュレーションを行なうためである。なお正常なセンサの場合、遅れ時間はゼロに設定されているため、問題なく動作する。

【0272】

図50は、干渉チェックによる関節可動範囲の検索法の説明図である。

【0273】

先ずユーザによって、どの方向の制限値を設定したいのかということと、どの部品とどの部品が接触することにより制限が起きるのかということが選択され、その後、関節可動範囲の検索が行なわれる。

【0274】

この関節可動範囲の検索においては、関節の姿勢 x を初期値 x_0 から Δx ずつ増加させ、その度に干渉チェックを行なう。すると図50のように、 x が $x_0 + n \Delta x$ では干渉が発生せず、次の $x_0 + (n + 1) \Delta x$ で干渉が起こる位置が判明する。続いて、 $\Delta x' = \Delta x / 10$ などのように Δx より小さい移動量 $\Delta x'$ を用いて同様に x を $x_0 + \Delta x$ から $\Delta x'$ ずつ増加させることにより、 x が $x_0 + n \Delta x + n' \Delta x'$ では干渉が発生せず、 $x_0 + n \Delta x + (n' + 1) \Delta x'$ で干渉が発生する位置が判明する。

【0275】

このような処理を数回繰り返すことにより、高速に、かつ求めようとする精度で x の制限値 $x_0 + n \Delta x + n' \Delta x' + n'' \Delta x'' \dots$ を求めることができる。ただしユーザが指定した二つの部品が、関節をどのように動かしても干渉しない位置関係であることも考えられるので、一回目の干渉位置検索の際には検索数nに上限を設け無限ループを防ぐことが好ましい。

【0276】

図51は、干渉チェックによるカム関係の検索方法の説明図、図52は、干渉チェックによるカム関係検索ルーチンのフローチャートである。

【0277】

ここでは、あらかじめ、ユーザにより、駆動部品、受動部品、駆動部品の初期姿勢、受動部品の初期姿勢、駆動部品の最終姿勢が指定される。

【0278】

図52に示すルーチンは、先ず、 ϕ 、 ϕ に、それぞれ、駆動部品初期姿勢、受動部品初期姿勢が設定され（ステップS2201）、次いで、受動部品の非接触姿勢が検索されて ϕ に格納される（ステップS2202）。

【0279】

ステップS2203では、今回求めた、駆動部品の姿勢 ϕ と、受動部品の非接触姿勢 ϕ とのペアが登録される。駆動部品の姿勢 ϕ が $\Delta \phi$ だけ移動され（ステップS2204）、駆動部品の姿勢 ϕ が最終姿勢になるまで、 $\Delta \phi$ ずつ移動された駆動部品に対する受動部品の非接触姿勢 ϕ の演算（ステップS2202）、登録（ステップS2203）が繰り返される。

【0280】

図53は、図52に示すカム関係検索ルーチンのステップS2202で実行さ

れる受動部品の非接触姿勢検索ルーチンのフローチャートである。

【0281】

先ずステップS2301において、受動部品の現在の姿勢 ϕ が ϕ_0 に設定され、カウンタとしてのnに初期値ゼロが設定される。

【0282】

ステップS2302では駆動部品と受動部品とが干渉しているか否かが判定され、干渉していないときは、現在の姿勢 ϕ が非接触姿勢となる（ステップS2303）。

【0283】

ステップS2302で干渉している旨判定されると、ステップS2303においてnがインクリメントされ、受動部品の姿勢 ϕ として $\phi_0 + n \Delta \phi$ が設定され（ステップS2304）、再度干渉しているか否かが判定される（ステップS2305）。まだ干渉していたときは、今度は受動部品の姿勢 ϕ として $\phi_0 - n \Delta \phi$ が設定され（ステップS2306）、干渉しているか否かがもう一度判定される（ステップS2307）。まだ干渉していたときは、ステップS2308に進み、nがnの上限か否か判定され、nの上限であったときは検索失敗とされる（ステップS2309）。nが未だ上限に達していなかったときは、ステップS2303に戻りnが1だけインクリメントされ、以下同様にして干渉チェックが行われる。

【0284】

ステップS2305で干渉していないと判定されると、ステップS2310に進み、 ϕ が $\Delta \phi$ だけデクリメントされ、 $\Delta \phi$ が適当な整数Nで割り算されて $\Delta \phi$ として小さな値が設定される（ステップS2303）。次いで、 ϕ が、上記のようにして小さな値に設定された $\Delta \phi$ だけインクリメントされながら（ステップS2313）、干渉しているか否かが判定され（ステップS2312）、干渉した状態から干渉していない状態に変化するとステップS2314に進んで所定の精度まで検索したか否かが判定され、未だ精度が不充分なときはステップS2310に戻って ϕ から $\Delta \phi$ が減算され、ステップS2311において $\Delta \phi$ としさらに小さい値が設定され、 ϕ をさらに小さい値である $\Delta \phi$ ずつインクリメントしながら干渉チェックが行なわれる（ステップS2312, S2313）。

【0285】

このようにして所定の精度まで検索が行なわれると（ステップS2314）、これにより非接触姿勢が定まる（ステップS2303）。

【0286】

ステップS2307において干渉していないと判定されたときもステップS2310～S2314の処理と同様の処理が実行され、受動部品の非接触姿勢が求められる。尚、ステップS2307において干渉していないと判定されたときは、ステップS2310に相当するステップにおいて ϕ は $\Delta \phi$ だけインクリメントされ、ステップS2313に相当するステップにおいて ϕ は $\Delta \phi$ だけデクリメントされる。

【0287】

図54、図52に示すカム関係検索ルーチンのステップS2202で実行される受動部品の非接触姿勢検索ルーチンのフローチャートである。このルーチンは、受動部品に重力あるいはバネ付勢力が作用している場合に実行されるルーチンである。受動部品に作用する重力あるいはバネの付勢方向はユーザによりあらかじめ設定される。図53は、負方向に力が加わっている場合のフローチャートを示している。

【0288】

図54のルーチンでは先ず、ステップS2401において、受動部品の現在の姿勢 ϕ が ϕ_0 に設定され、カウンタとしてのnに初期値ゼロが設定される。

【0289】

ステップS2402では、駆動部品と受動部品が現在干渉しているか否かが判

定される。既に干渉しているときは、駆動部品の今回の姿勢変化によって受動部品が干渉したものであるため、受動部品の姿勢を重力あるいはバネ付勢力にからって変化させ（ステップS 2404～S 2407）、干渉していない姿勢をつけた後、 ϕ から $\Delta\phi$ を減算することで（ステップS 2408）、一旦干渉した状態に戻す。

【0290】

一方、ステップS 2402で干渉していない旨判定されるときは、駆動部品の今回の姿勢変化によって受動部品が干渉していない状態に変化したものであるため、重力あるいはバネ付勢力の作用をシミュレートして受動部品が能動部品に干渉した状態を見つける（ステップS 2409～S 2413）。

【0291】

ステップS 2414に進んだとき、 $\Delta\phi$ のきざみでぎりぎり干渉した状態にあり、 $\Delta\phi$ をNで割り算することにより、 $\Delta\phi$ として小さな値を設定し、 ϕ を $\Delta\phi$ ずつインクリメントしながら非干渉姿勢が検索される（ステップS 2415、S 2416）。ステップS 2417では所定の精度まで検索が行なわれたか否かが判定され、未だ精度が不充分なときはステップS 2408に進んで ϕ から $\Delta\phi$ が減算されて干渉した状態に戻され、ステップS 2414において $\Delta\phi$ がNで割り算されて $\Delta\phi$ としてさらに小さい値が設定されて検索が繰り返される。

【0292】

このようにして所定の精度まで検索が行なわれると（ステップS 2414）、これにより非接触位置が決定される（ステップS 2418）。

【0293】

このようにして、重力あるいはバネ付勢力が作用しない場合であっても作用する場合であっても、カム関係を決定することができる。

【0294】

図55は、溝にピンが挿入されている溝関係を設定する手法の説明図である。

【0295】

図55に示すような溝関係を設定しようとしたとき、ピンは常に接触した状態にあるため、非干渉位置を求ることはできず、このような溝関係を前述したカム関係の検索法を用いて求ることは困難である。

【0296】

そこで、ここでは、図55（c）に示すように、ピンの中心に半径 r の微小な円筒を一時的に取り付け、この円筒と溝との間の距離 d を測定することで溝関係を求める。すなわち、図53における「干渉しているか？」の判定に代えて「 $d < R - r$ か？」を判定することで溝関係を検索することができる。

【0297】

【発明の効果】

以上説明したように、本発明によれば、機構を物理的に製作することなく、機構制御用のプログラムのデバックを行なうことができ、その機構制御プログラムの開発の短縮化、コスト低減化に大きく寄与することになる。

【図面の簡単な説明】

【図1】

内部に機構制御プログラム開発支援システムが構築されたコンピュータシステムの外観図である。

【図2】

図1に外観を示すコンピュータシステムのハードウェア構成図である。

【図3】

プログラム記憶媒体内に記憶されたプログラムの構成を示す図である。

【図4】

機構制御プログラム開発支援システムの一例が構築されたコンピュータネットワークの一例を示す図である。

【図5】

図4に外観を示すコンピュータネットワークを構成するコンピュータシステムのハードウェア構成図である。

【図6】

機構制御プログラム開発支援システムの概念図である。

【図7】

センサやモータの定義時の処理を示す3Dモデルシミュレータの模式図である

【図8】

シミュレーション時の3Dモデルシミュレータの処理を示す模式図である。

【図9】

シミュレーション時における3Dモデルシミュレータと制御プログラムの動作タイミングチャートを示す図である。

【図10】

シミュレーション時のシミュレータのフローを示すフローチャートである。

【図11】

図10に示すシミュレータの処理フローの一部分(図10に一点鎖線で囲った部分)に代えて採用することのできる部分フローを示すフローチャートである。

【図12】

図12は、モータとして定義されるリンクの選び方を示す模式図である。

【図13】

センサとして定義されるリンクの選び方を示す模式図である。

【図14】

センサとして定義されるリンクの選び方を示す模式図である。

【図15】

モータの種類を示す模式図である。

【図16】

モータを一次遅れ系とみなしたときの、モータの回転速度 ω の初期変化を示す図である。

【図17】

シミュレータにおける、一次遅れを考慮したモータ駆動の処理を示すフローチャートである。

【図18】

干渉チェックを用いたセンサの定義方法の説明図である。

【図19】

干渉チェックを用いたセンサの定義方法の説明図である。

【図20】

干渉チェックを用いたセンサの定義方法の説明図である。

【図21】

シミュレータと制御プログラムとの間で同期をとるために送受信される信号のタイミングチャートである。

【図22】

3Dモデルの動作との同期をとるための同期化プログラムを示すフローチャートである。

【図23】

同期を実現するための、シミュレータ側の処理を示すフローチャートである。

【図24】

干渉可能性の評価方法の説明図である。

【図25】

部品間の距離 d とシミュレーション間隔 Δt との関係の各種の例を示す図である。

【図26】

部品のグループ分けの説明図である。

【図27】

1つの3Dモデル内に複数のモータリンクが定義されている場合のグループ分けの説明図である。

【図28】

グループ分けした部品の再グループ化の説明図である。

【図29】

モータリンクの動きによって姿勢に影響を受けるリンクのグループとそのモータリンクの動きによっては姿勢に影響を受けないグループとに分けるルーチンを示すフローチャートである。

【図30】

検索ルーチンのフローチャートである。

【図31】

図29, 図30に示すルーチンの動作説明用のリンク機構モデルを示す図である。

【図32】

図31に示すリンク機構モデルのデータ構造を示す図である。

【図33】

1つのモータリンクの動きによって姿勢の変化を受けるグループ内での干渉可能性の評価方法の説明図である。

【図34】

モータリンクの動きによって姿勢に影響を受けるリンクグループ内で、シミュレーション時に距離を求めるリンクの組を抽出する手法の説明図である。

【図35】

本実施形態におけるシミュレーション全体の処理の流れを示す図である。

【図36】

モータ処理ルーチンを表わすフローチャートである。

【図37】

リンク姿勢移動サブルーチンのフローチャートである。

【図38】

関節不良動作処理ルーチンのフローチャートである。

【図39】

センサ処理ルーチンのフローチャートである。

【図40】

オン/オフセンサ処理ルーチンのフローチャートである。

【図41】

センサ性能劣化処理ルーチンのフローチャートである。

【図42】

チャッタリングによる出力予定値変更処理ルーチンのフローチャートである。

【図43】

エンコーダの変位(角度等)に対する出力波形の一例を示す図である。

【図44】

エンコーダの、各種の出力波形を示す図である。

【図45】

エンコーダの出力処理ルーチンのプログラムを示す図である。

【図46】

エンコーダ処理ルーチンのフローチャートである。

【図47】

ポテンショメータの関数値の一例を示す図である。

【図48】

ポテンショメータ前処理ルーチンのフローチャートである。

【図49】

ΔT 決定処理ルーチンのフローチャートである。

【図50】

干渉チェックによる関節可動範囲の検索法の説明図である。

【図51】

干渉チェックによるカム関係の検索方法の説明図である。

【図52】

干渉チェックによるカム関係検索ルーチンのフローチャートである。

【図53】

受動部品の非接触姿勢検索ルーチンのフローチャートである。

【図54】

受動部品の非接触姿勢検索ルーチンのフローチャートである。

【図55】

溝にピンが挿入されている溝関係を設定する手法の説明図である。

【符号の説明】

1 0 0, 4 0 0, 5 0 0	コンピュータシステム
1 0 1, 4 0 1, 5 0 1	本体部
1 0 1 a, 4 0 1 a, 5 0 1 a	フロッピィディスク装填口
1 0 1 b, 4 0 1 b, 5 0 1 b	CDROM装填口
1 0 2, 4 0 2, 5 0 2	CRTディスプレイ
1 0 2 a, 4 0 2 a, 5 0 2 a	表示画面
1 0 3, 4 0 3, 5 0 3	キーボード
1 0 4, 4 0 4, 5 0 4	マウス
2 1 0, 4 1 0	CDROM
2 1 1, 4 1 1	ハードディスク
2 1 2, 4 1 2	フロッピィディスク
2 2 0, 4 2 0	バス
2 2 1, 4 2 1	中央演算処理装置 (CPU)
2 2 2, 4 2 2	RAM
2 2 3, 4 2 3	ハードディスクコントローラ
2 2 4, 4 2 4	フロッピィディスクドライバ
2 2 5, 4 2 5	CDROMドライバ
2 2 6, 4 2 6	マウスコントローラ
2 2 7, 4 2 7	キーボードコントローラ
2 2 8, 4 2 8	ディスプレイコントローラ
3 0 0	プログラム記憶媒体
3 1 0	機構制御プログラム開発支援プログラム
3 1 1	三次元機構モデルシミュレーションプログラム
3 1 2	同期化プログラム
4 2 9	モデム

【書類名】 要約書

【要約】

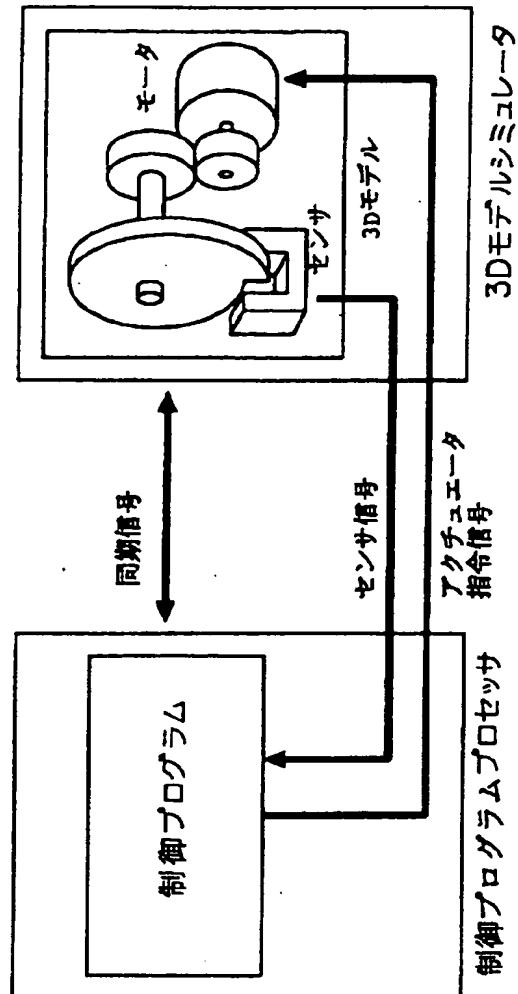
【課題】 機構を制御する制御プログラムの開発を支援する。

【解決手段】 内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータと、三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行する制御プログラムプロセッサとを備える。

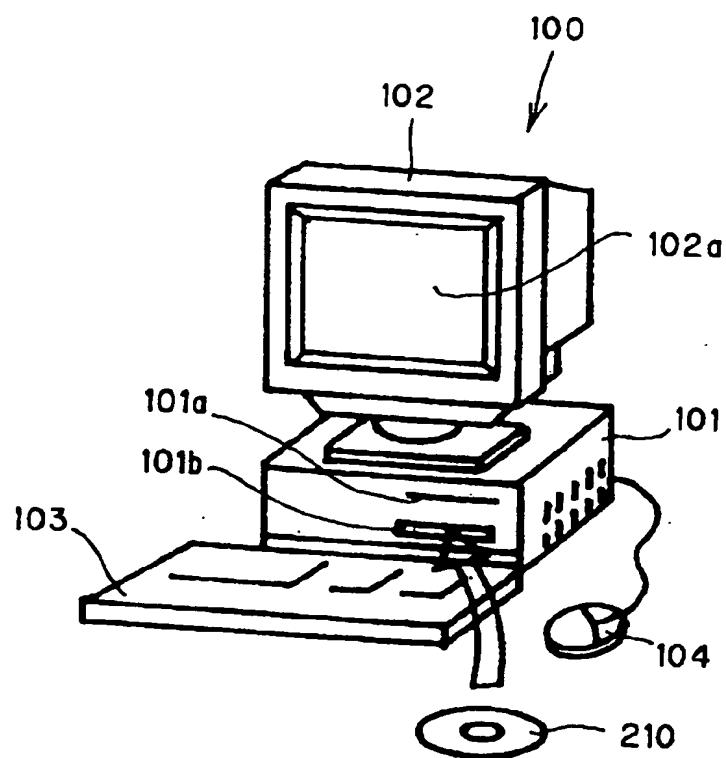
【選択図】 図 6

【書類名】
【図6】

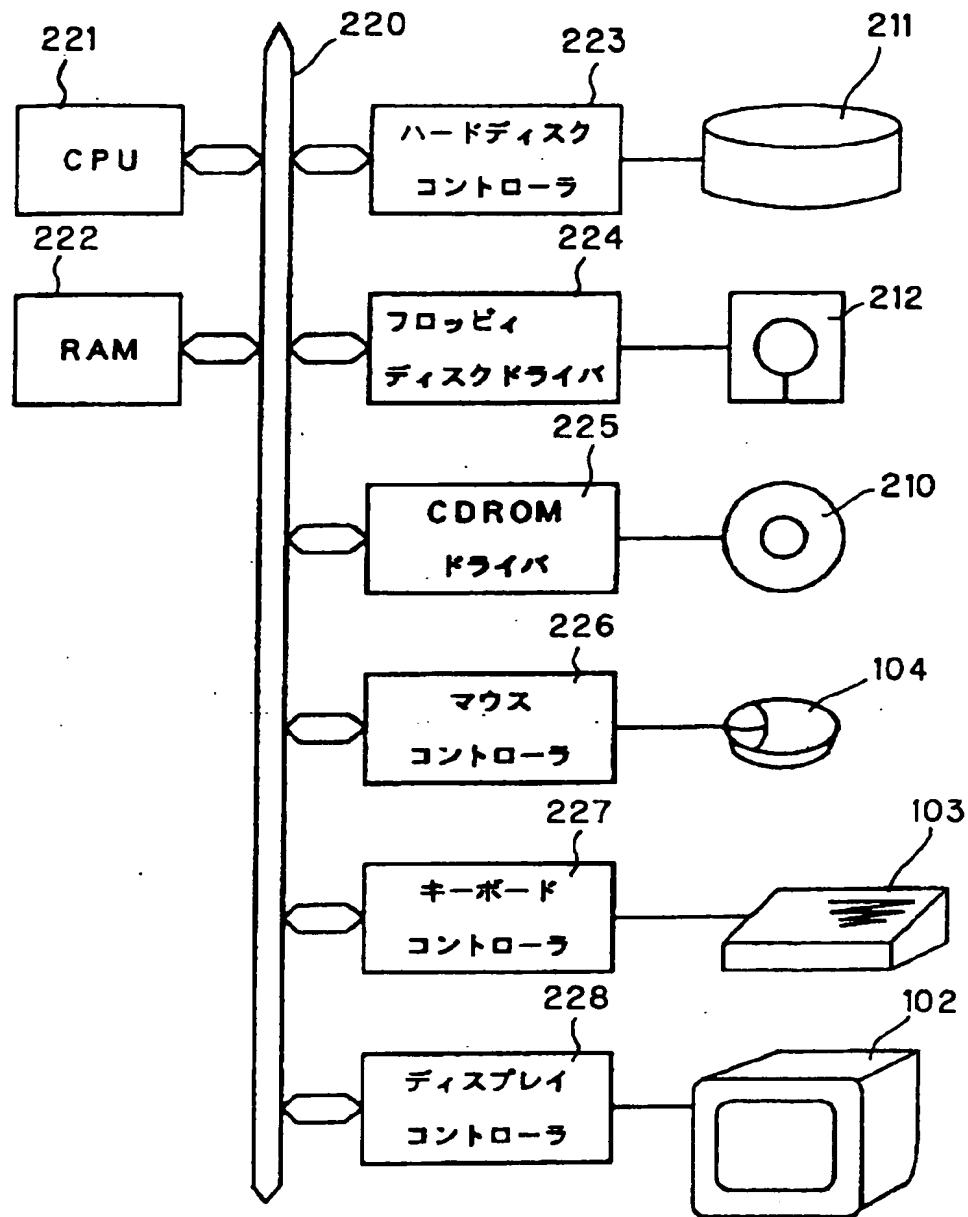
図面



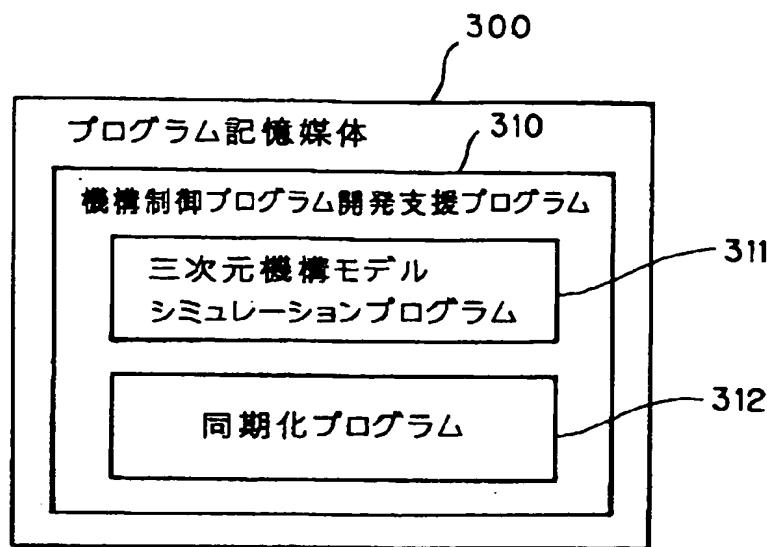
【図1】



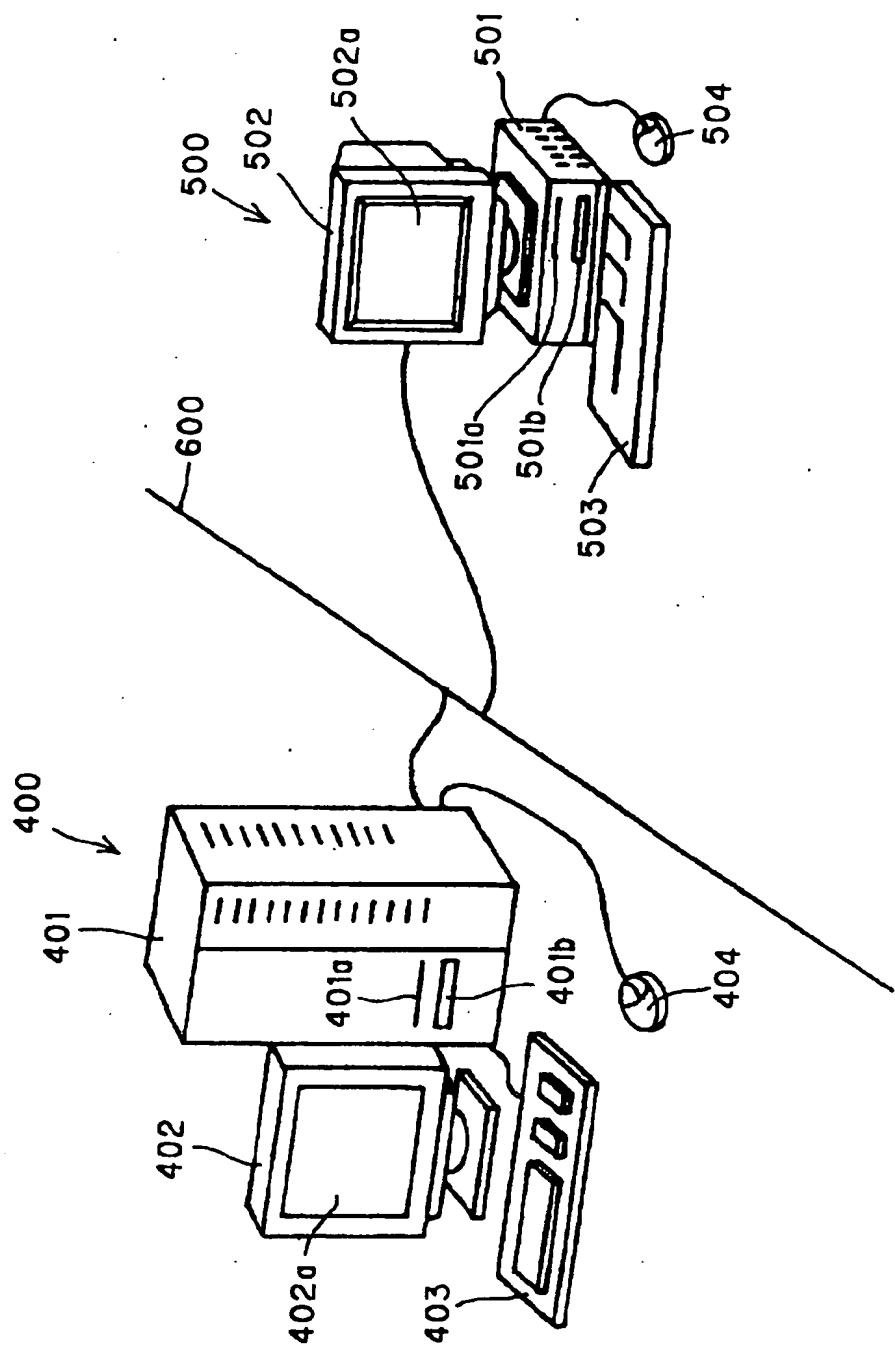
【図2】



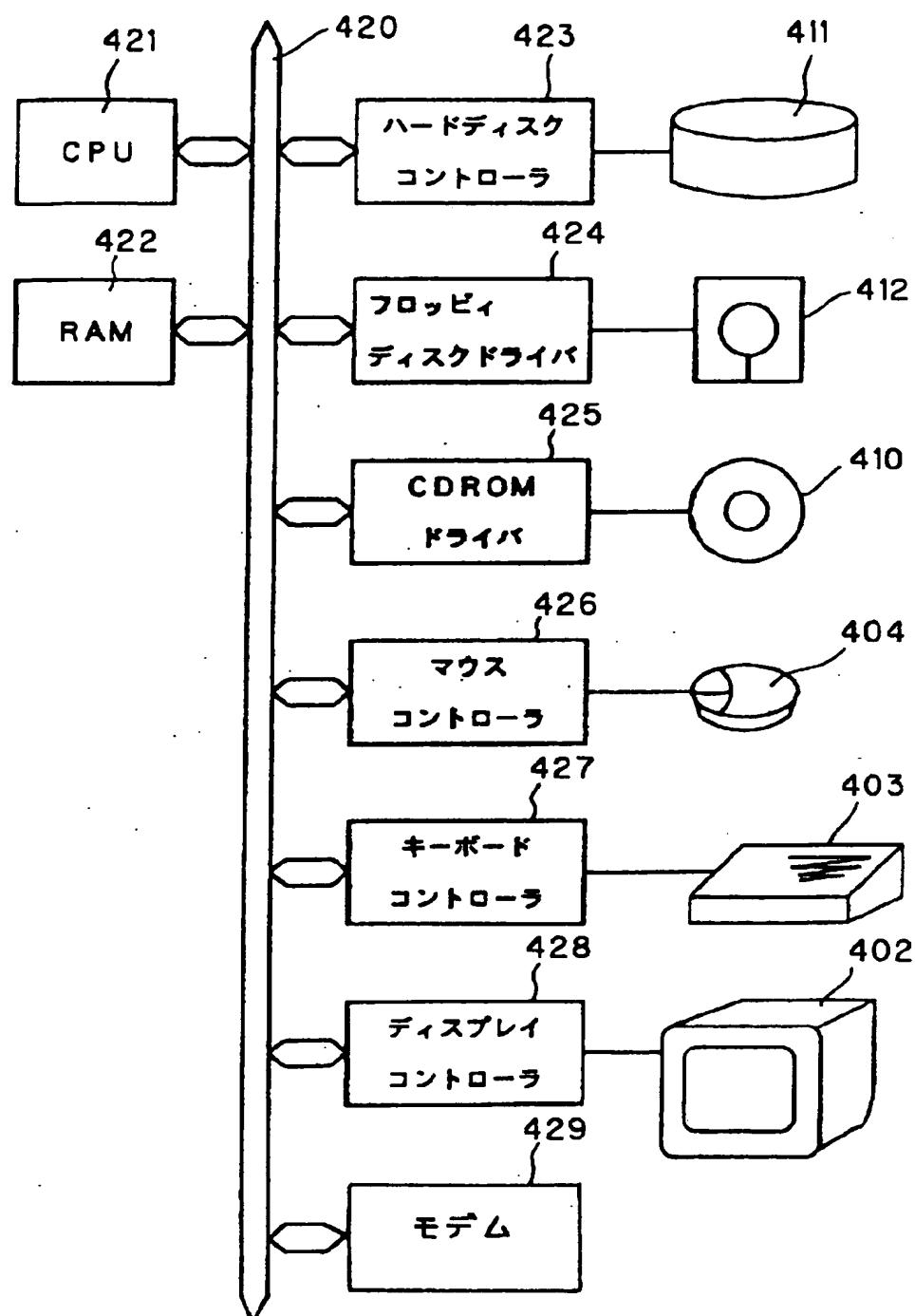
【図3】



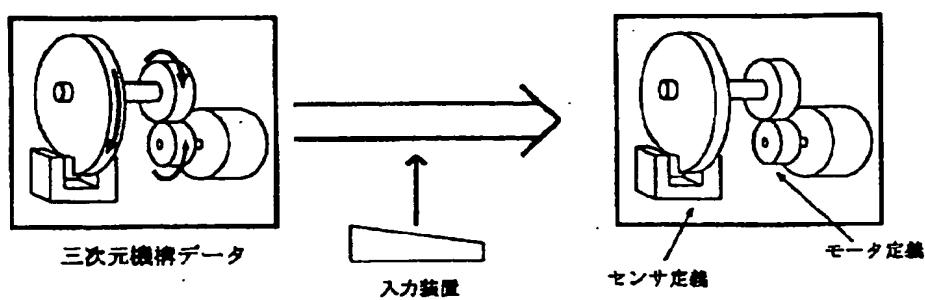
【図4】



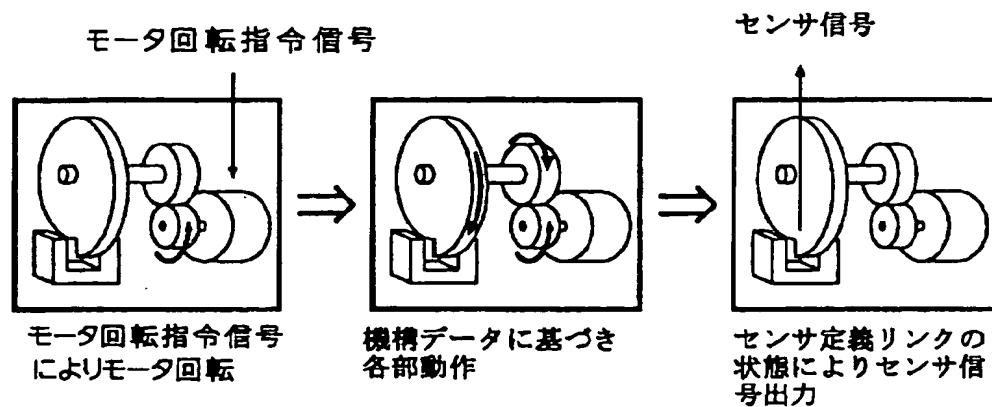
【図5】



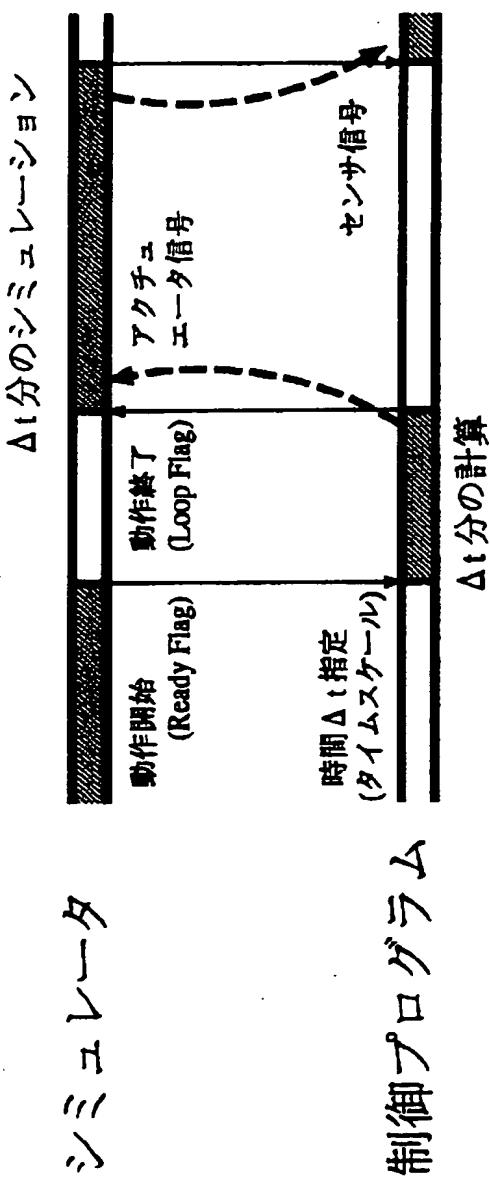
【図7】



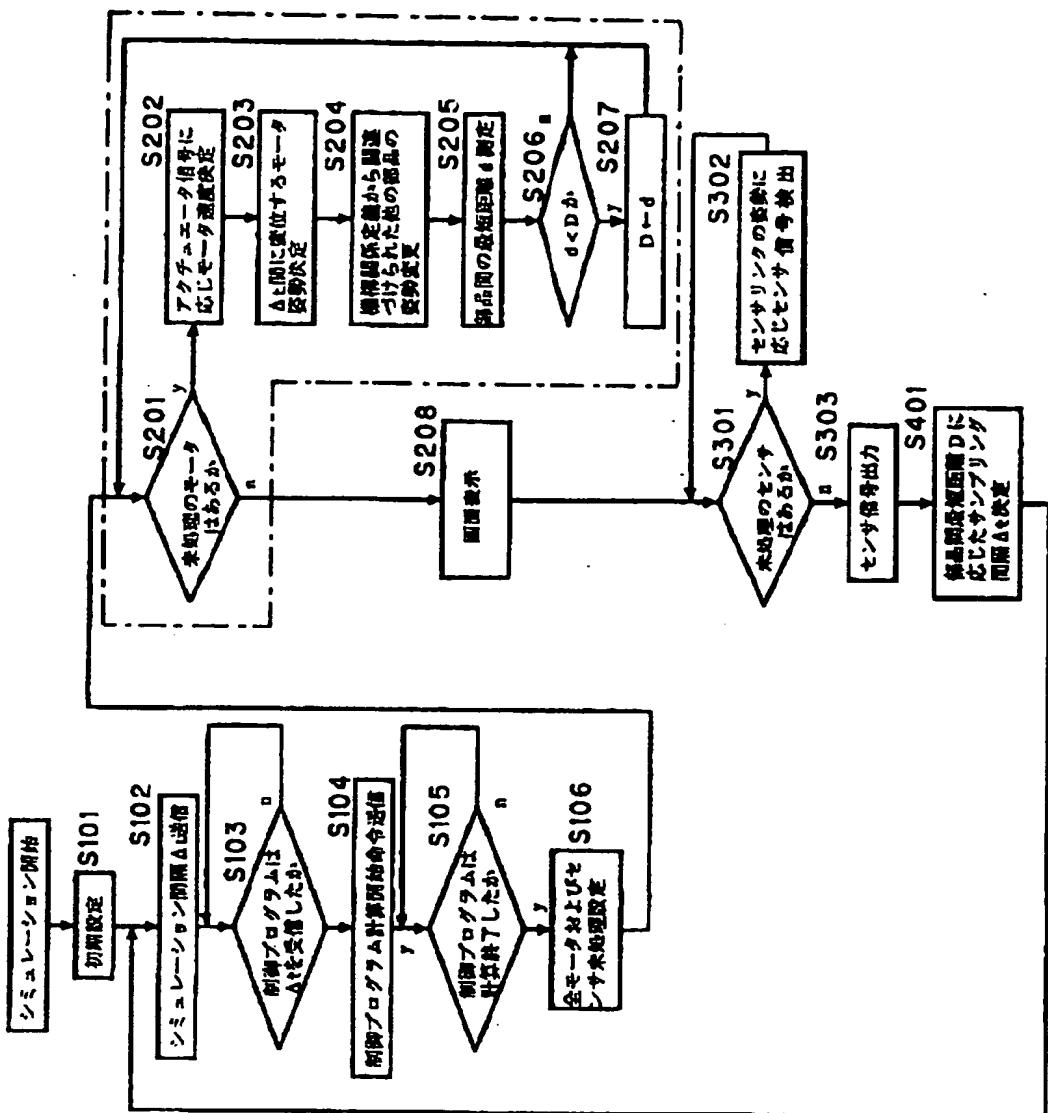
【図8】



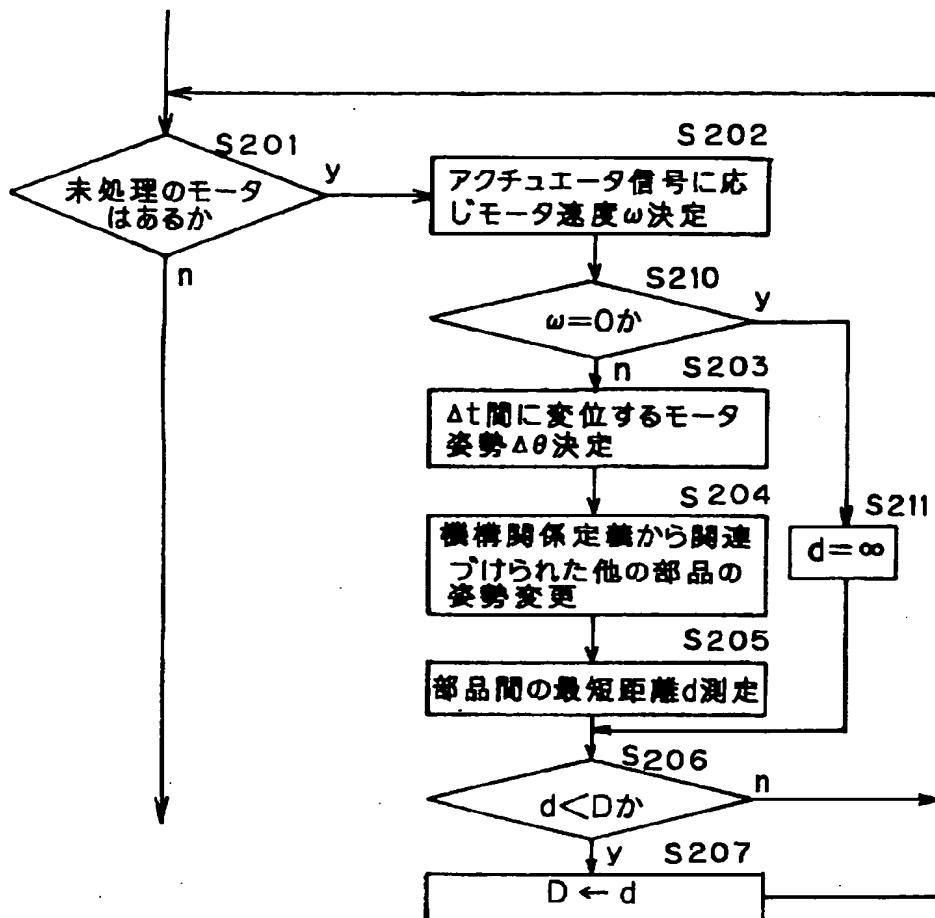
【図9】



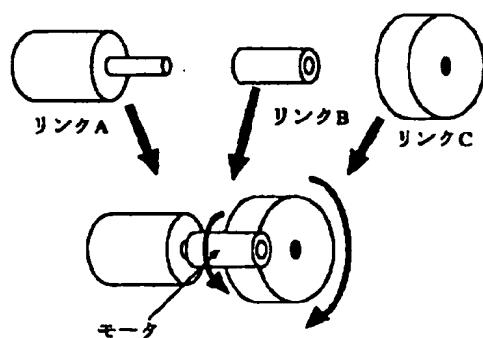
【図10】



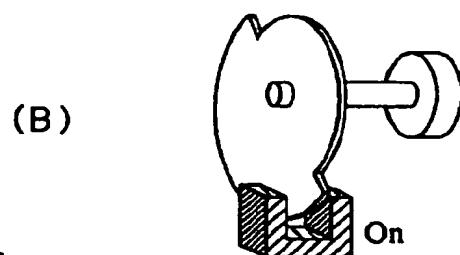
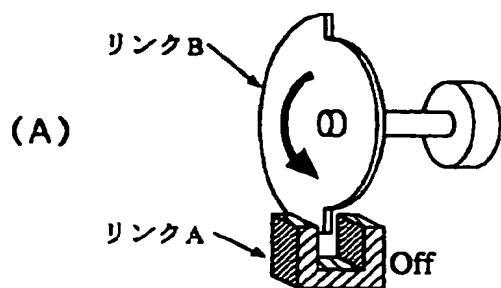
【図11】



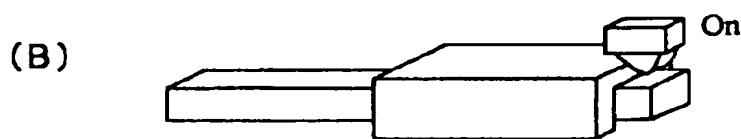
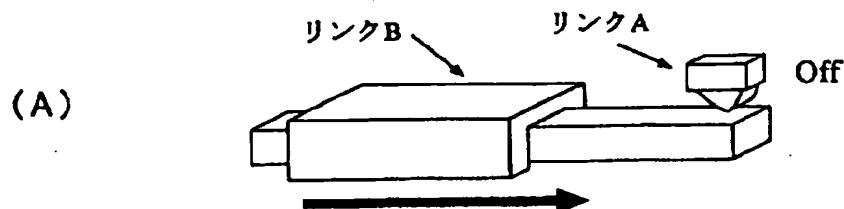
【図12】



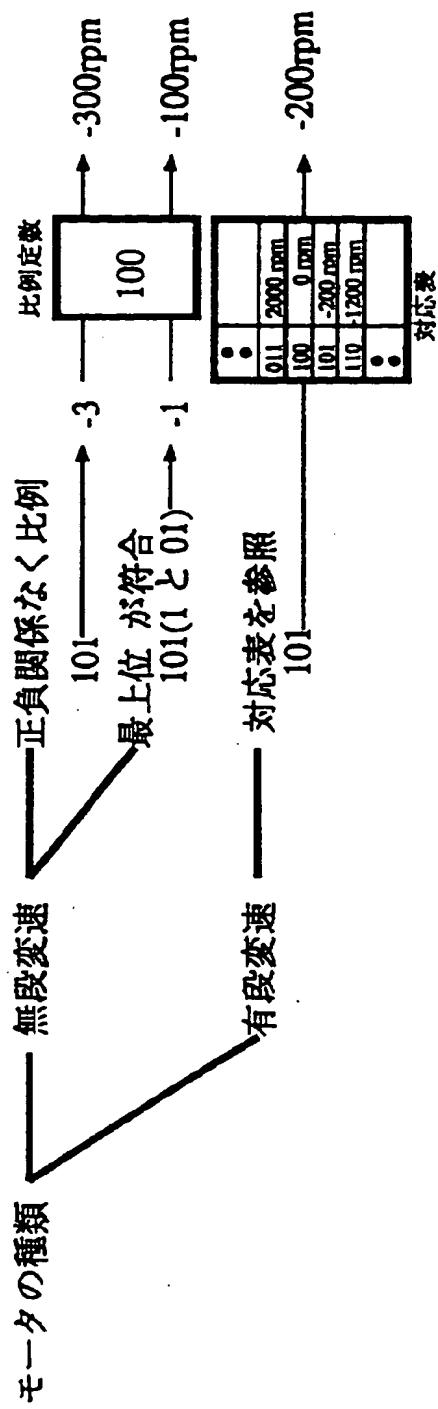
【図13】



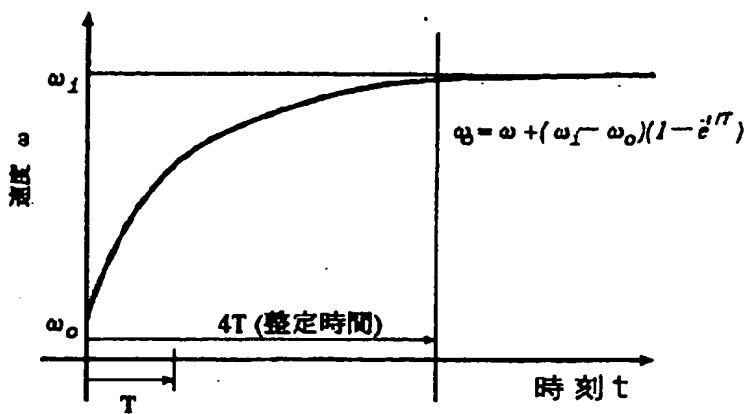
【図14】



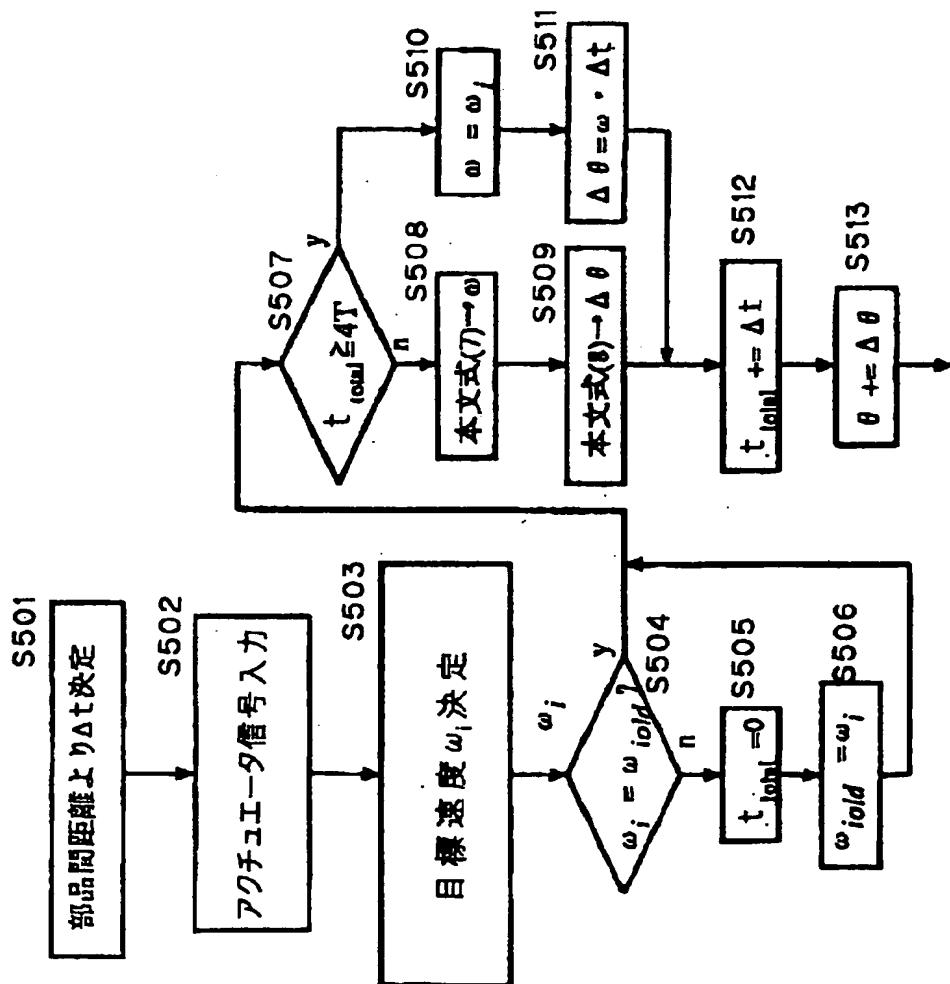
【図 15】



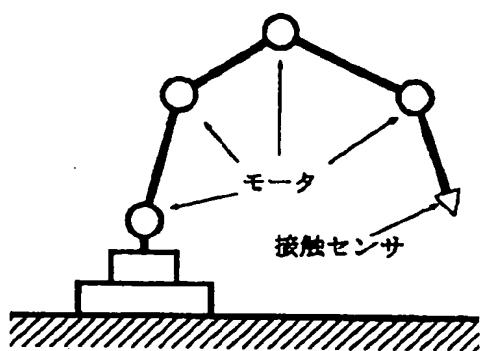
【図16】



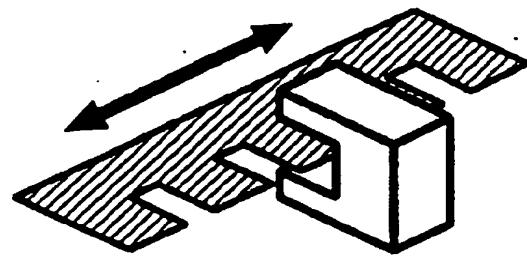
【図17】



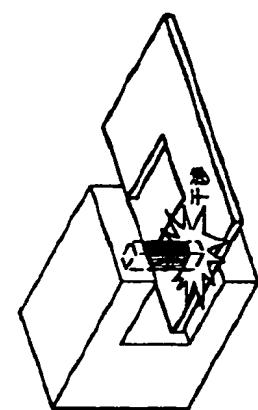
【図18】



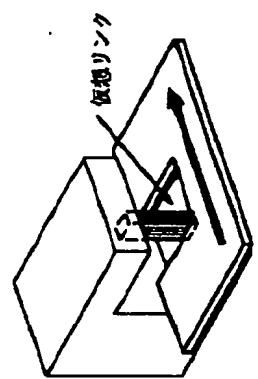
【図19】



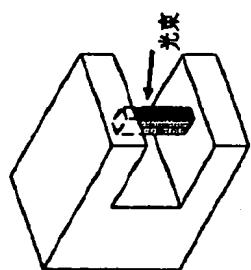
【図20】



(C)

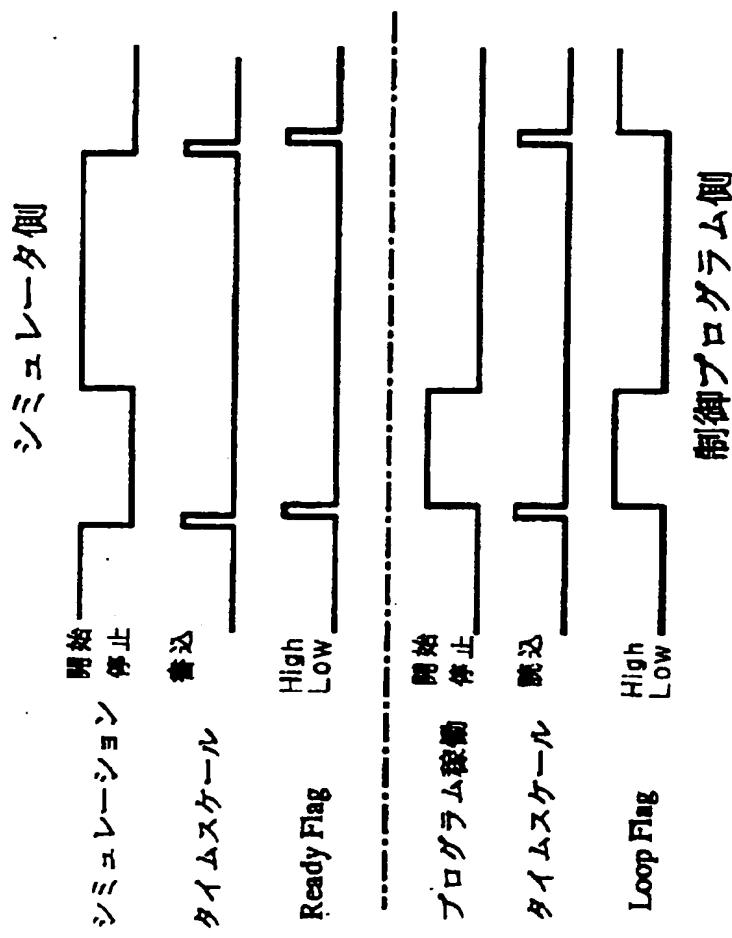


(B)

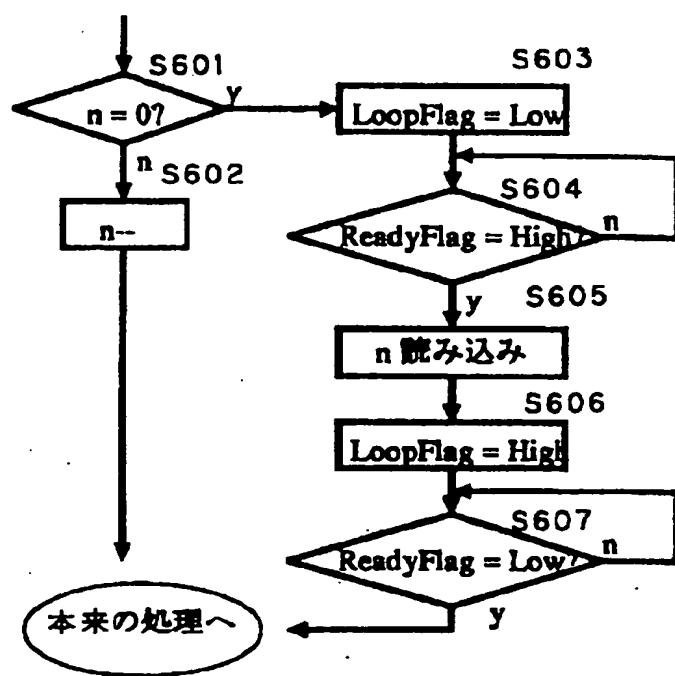


(A)

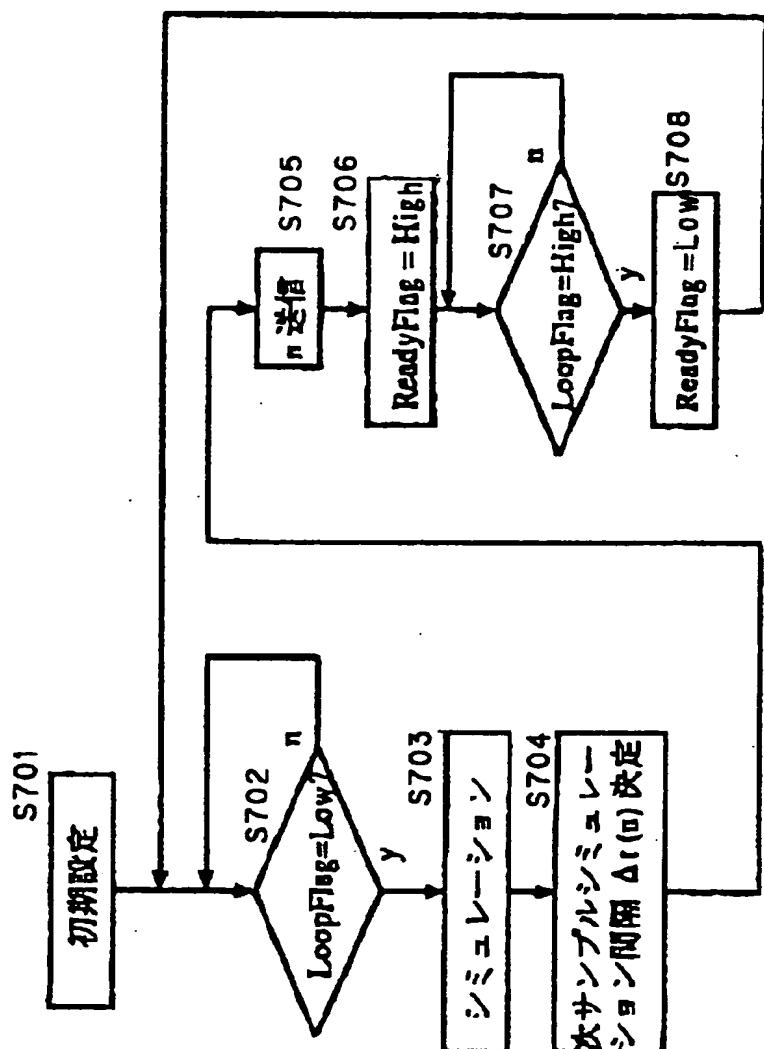
【図21】



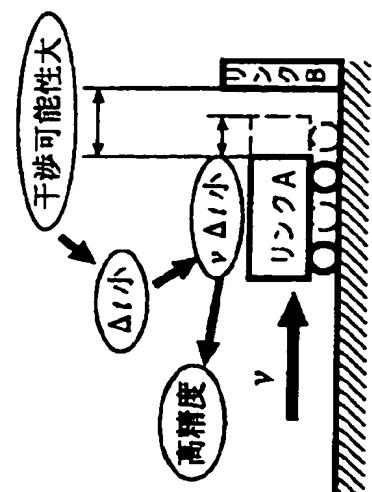
【図22】



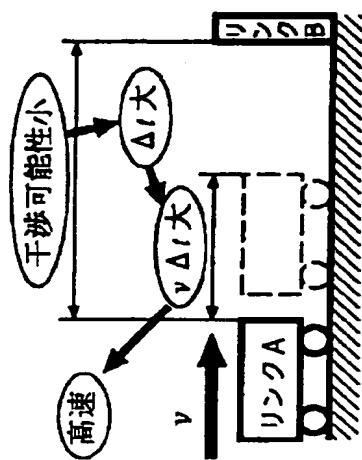
【図23】



【図24】

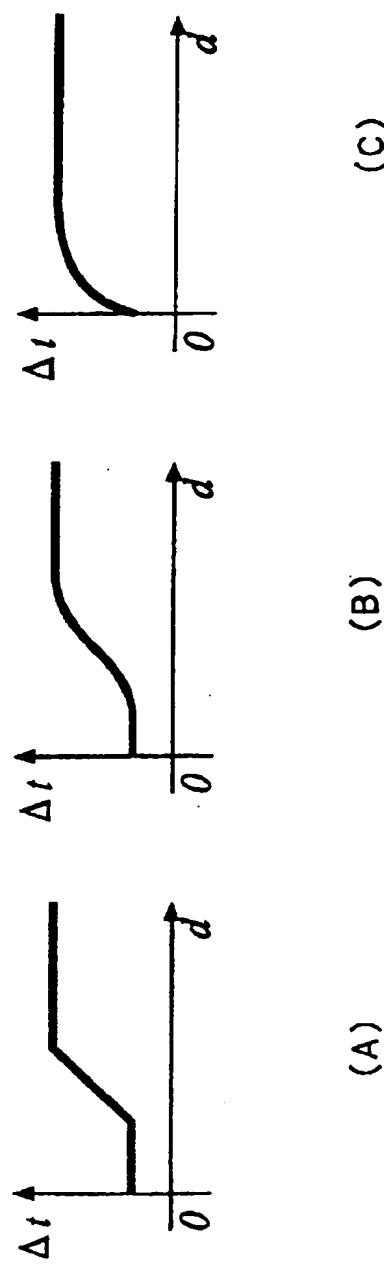


(B)

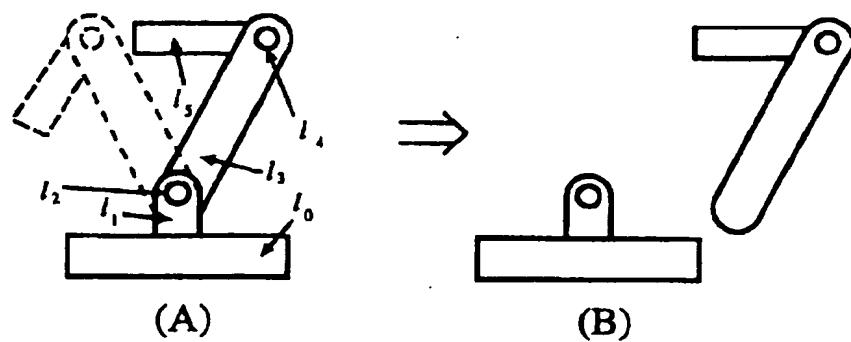


(A)

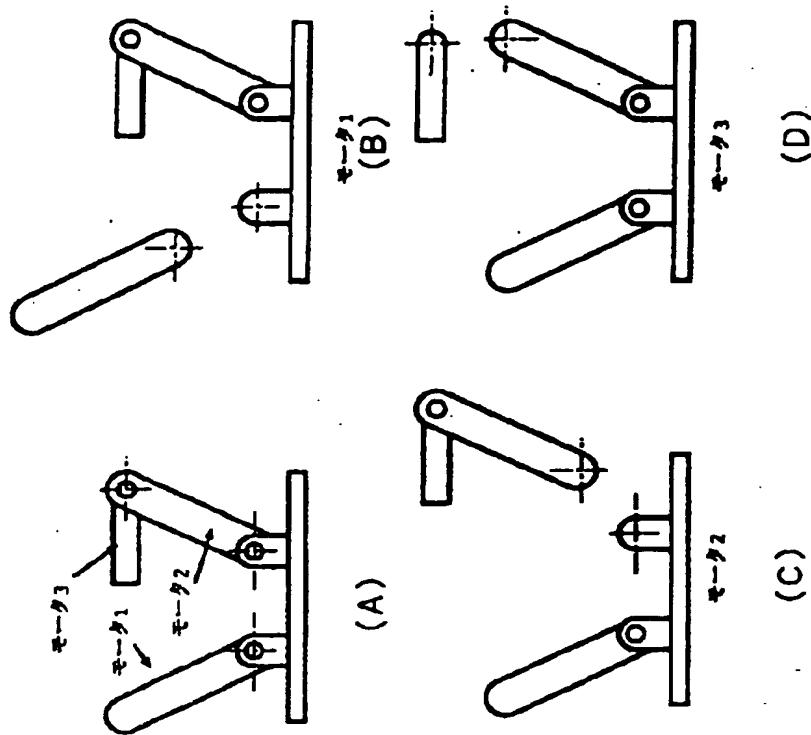
【図25】



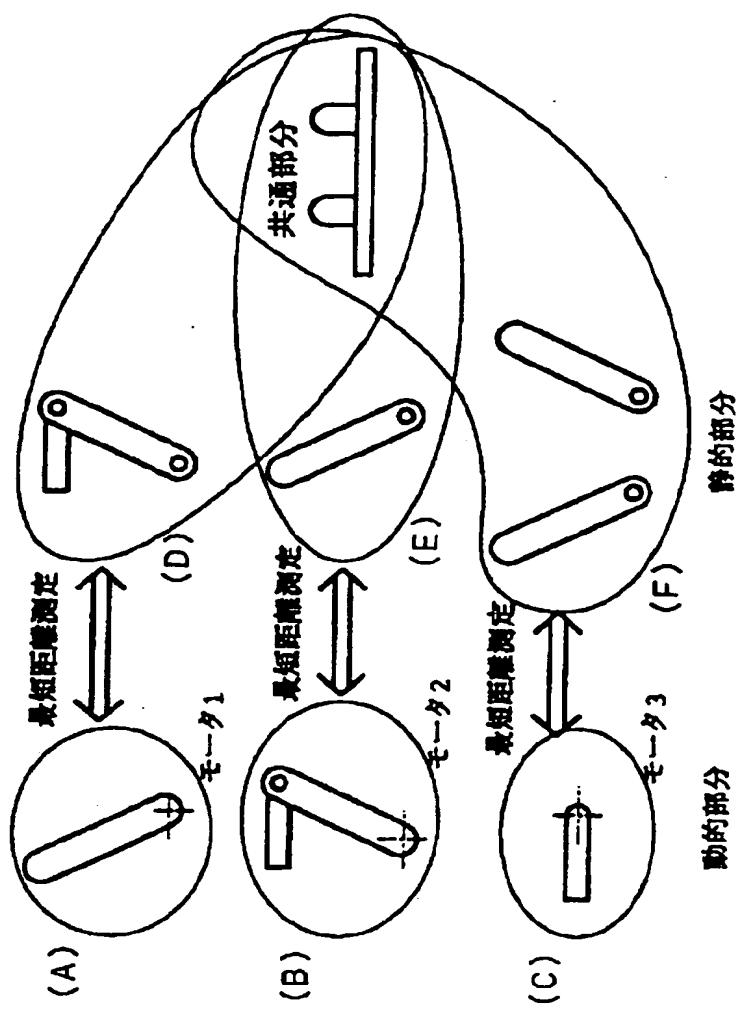
【図26】



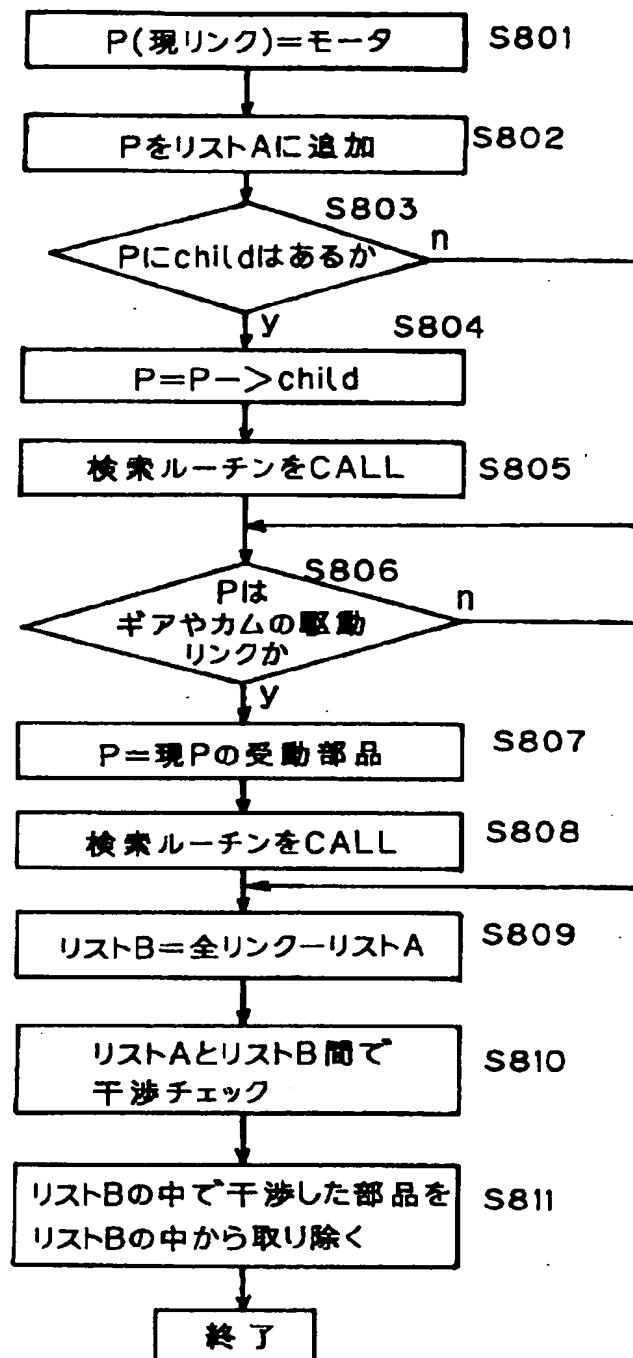
【図27】



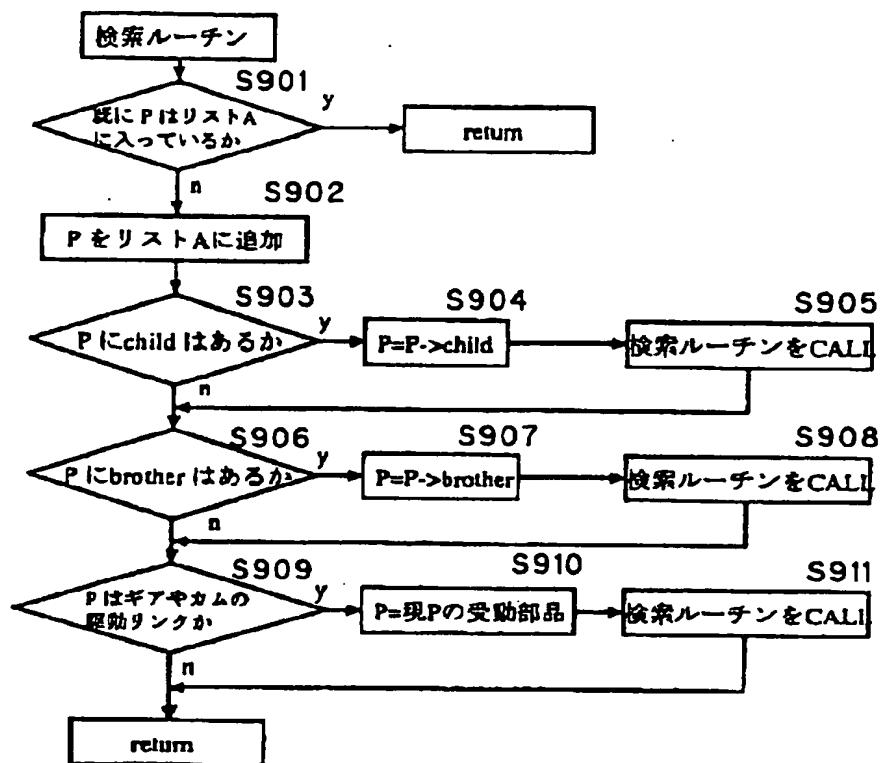
【図28】



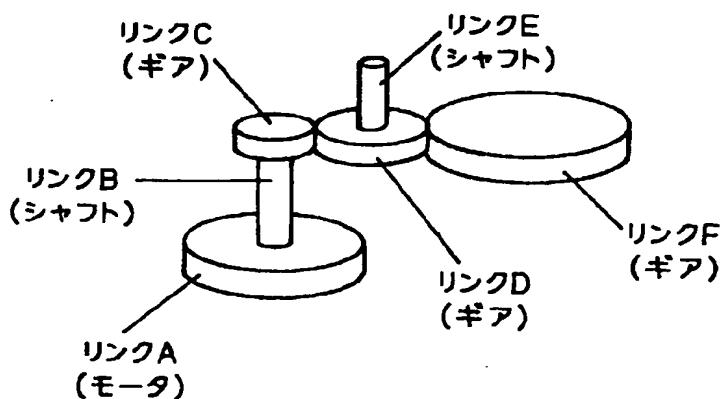
【図29】



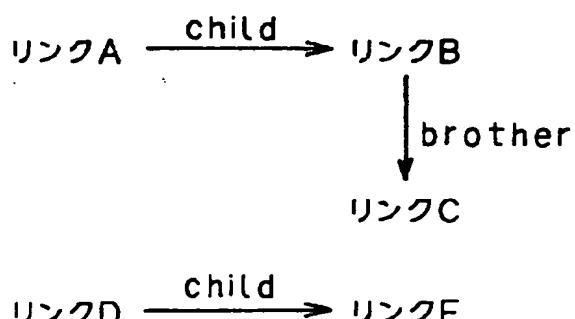
【図30】



【図31】

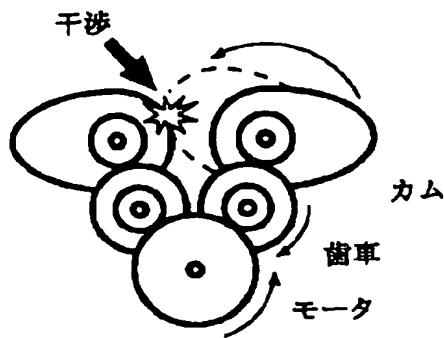


【図32】

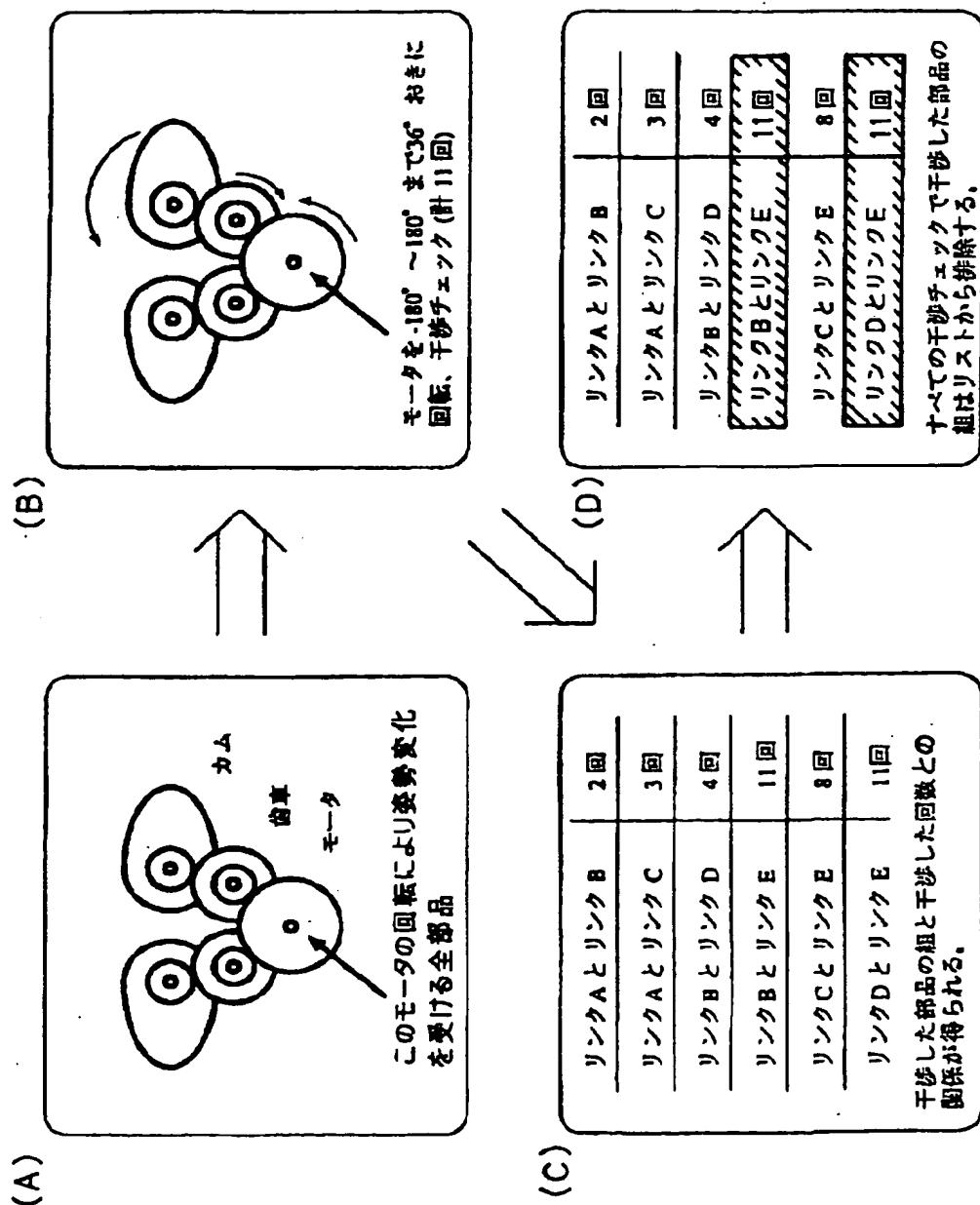


リンクF

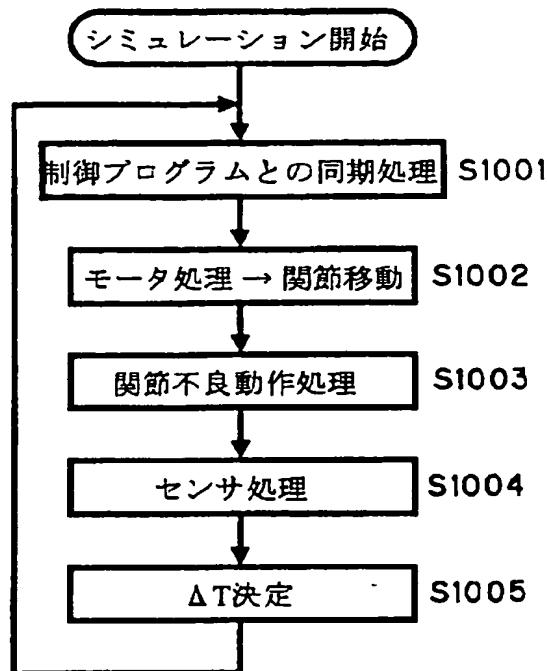
【図33】



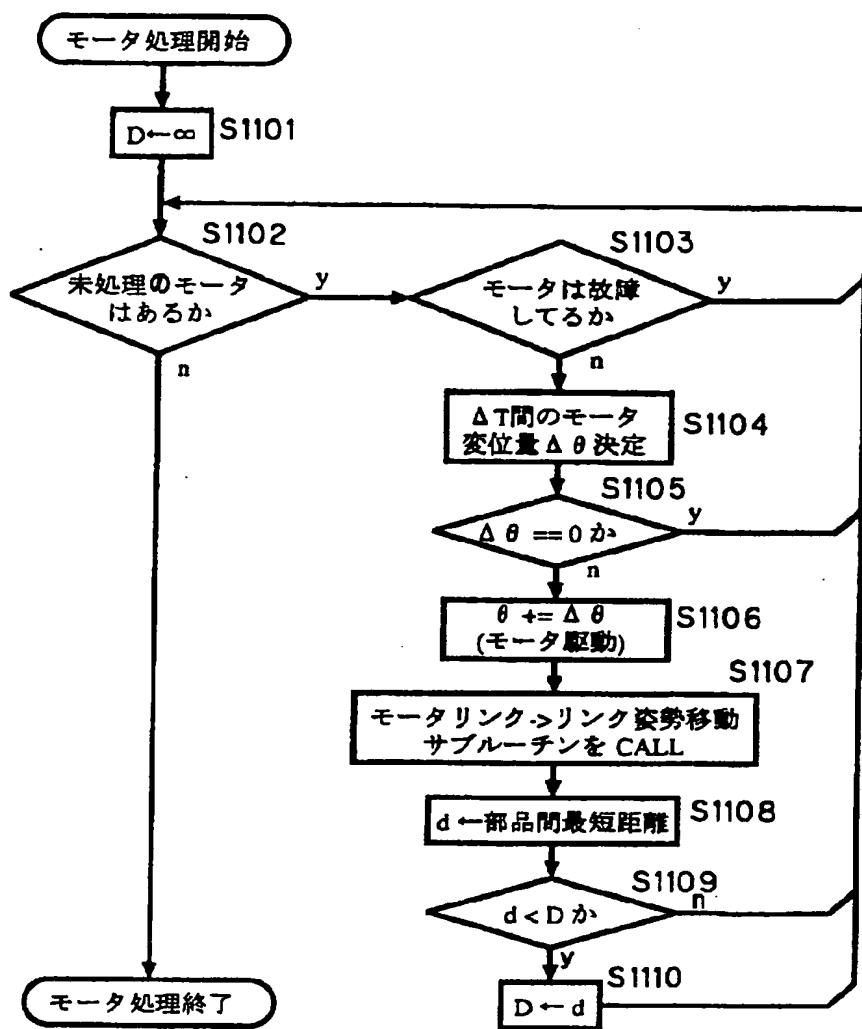
【図34】



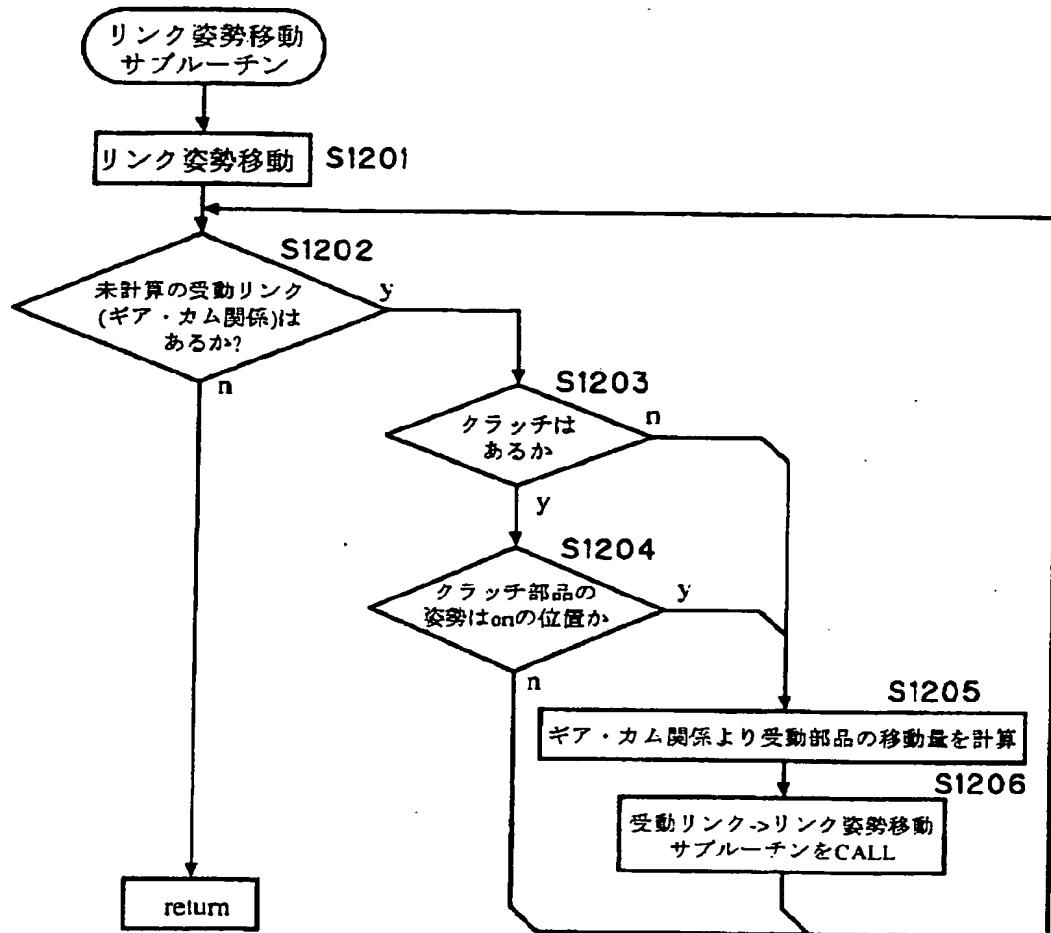
【図35】



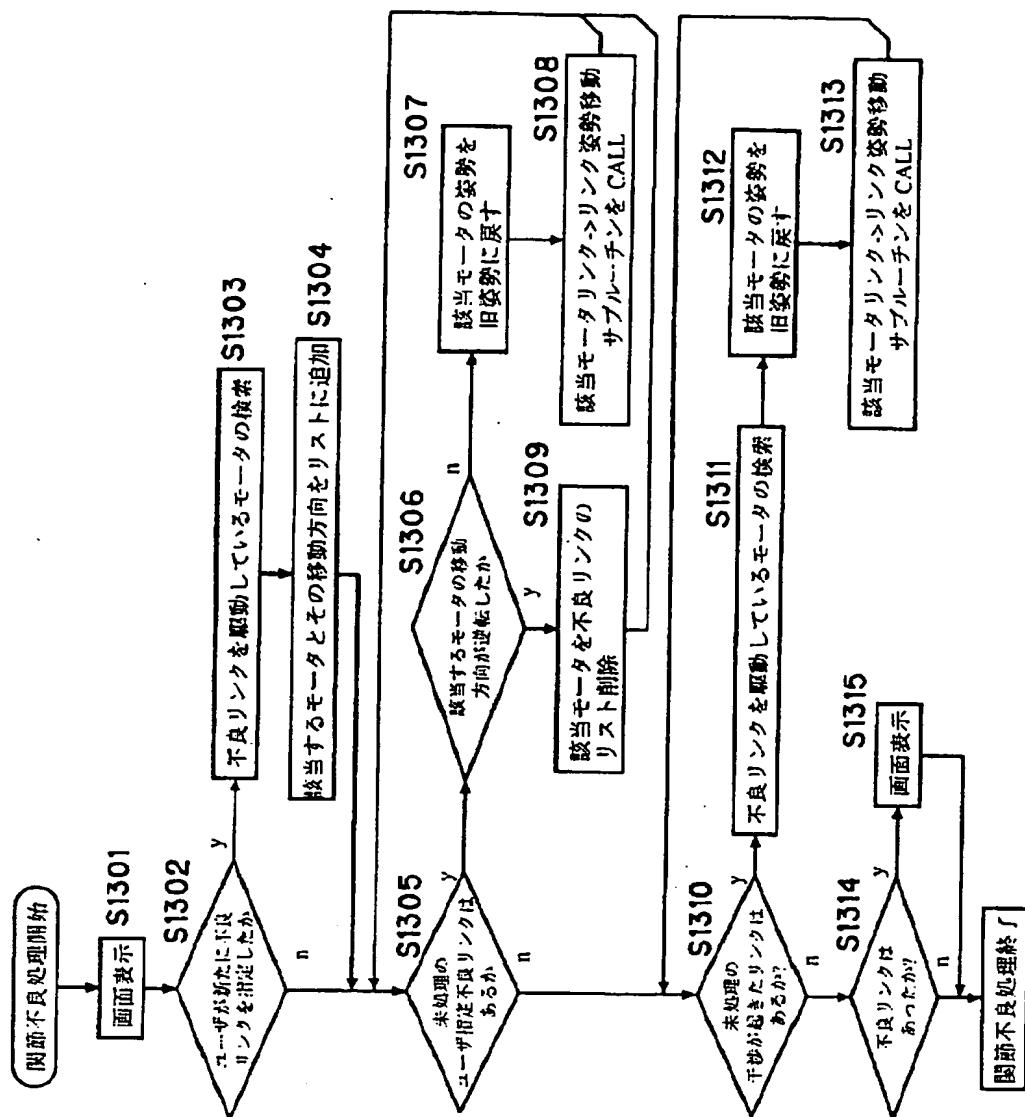
【図36】



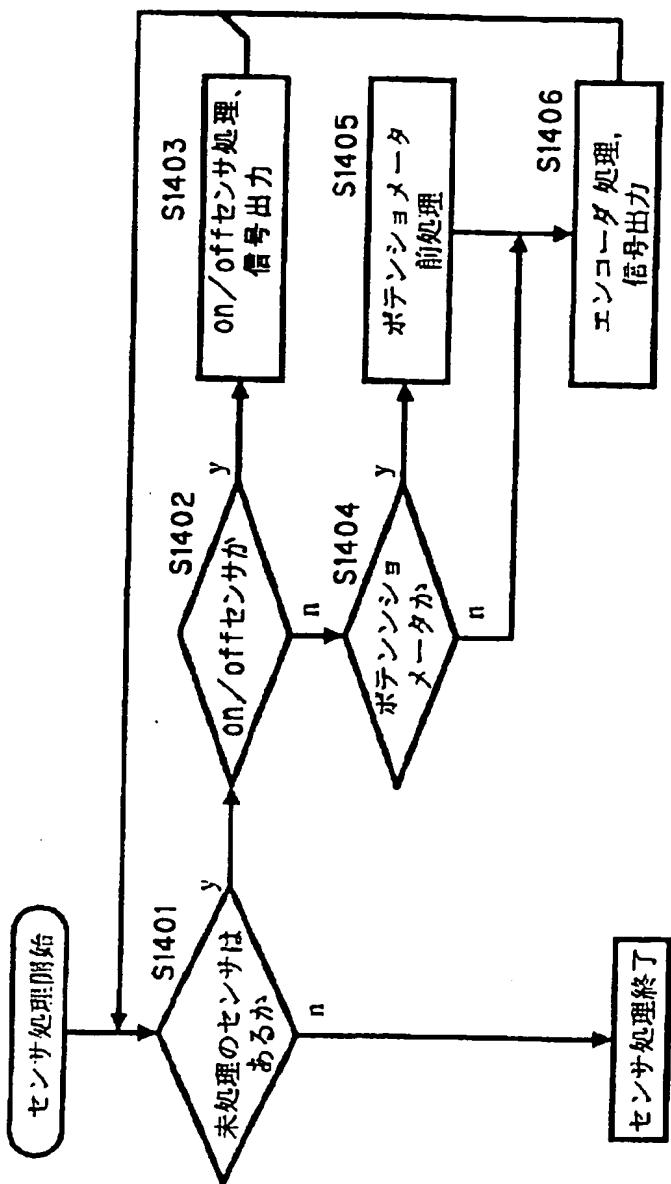
【図37】



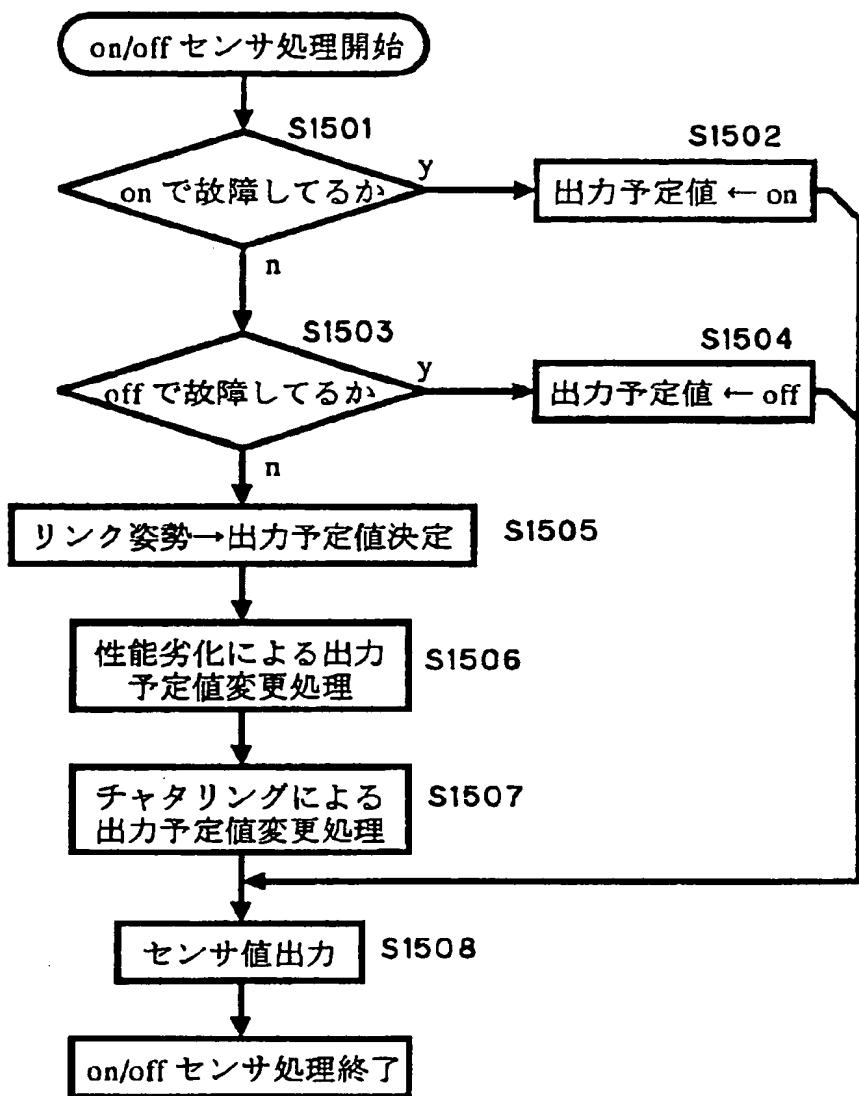
【図38】



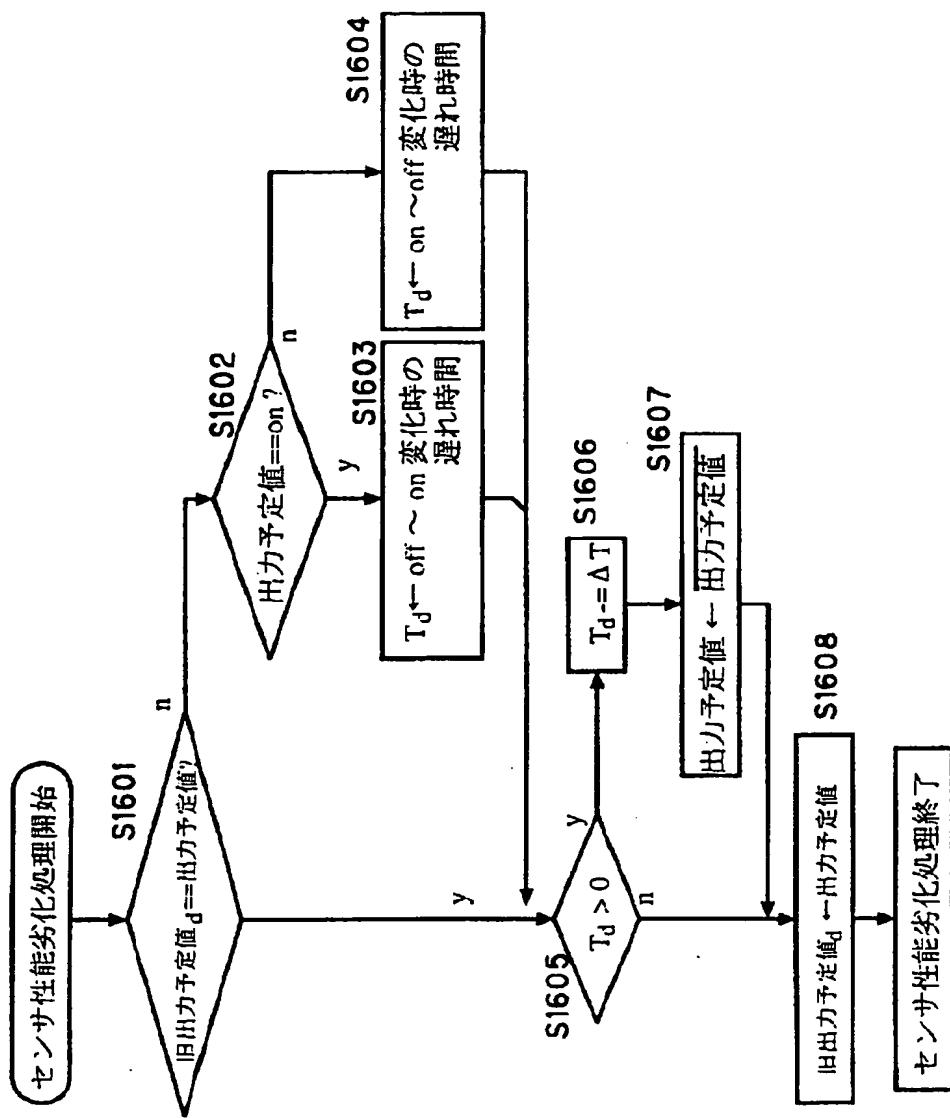
【図39】



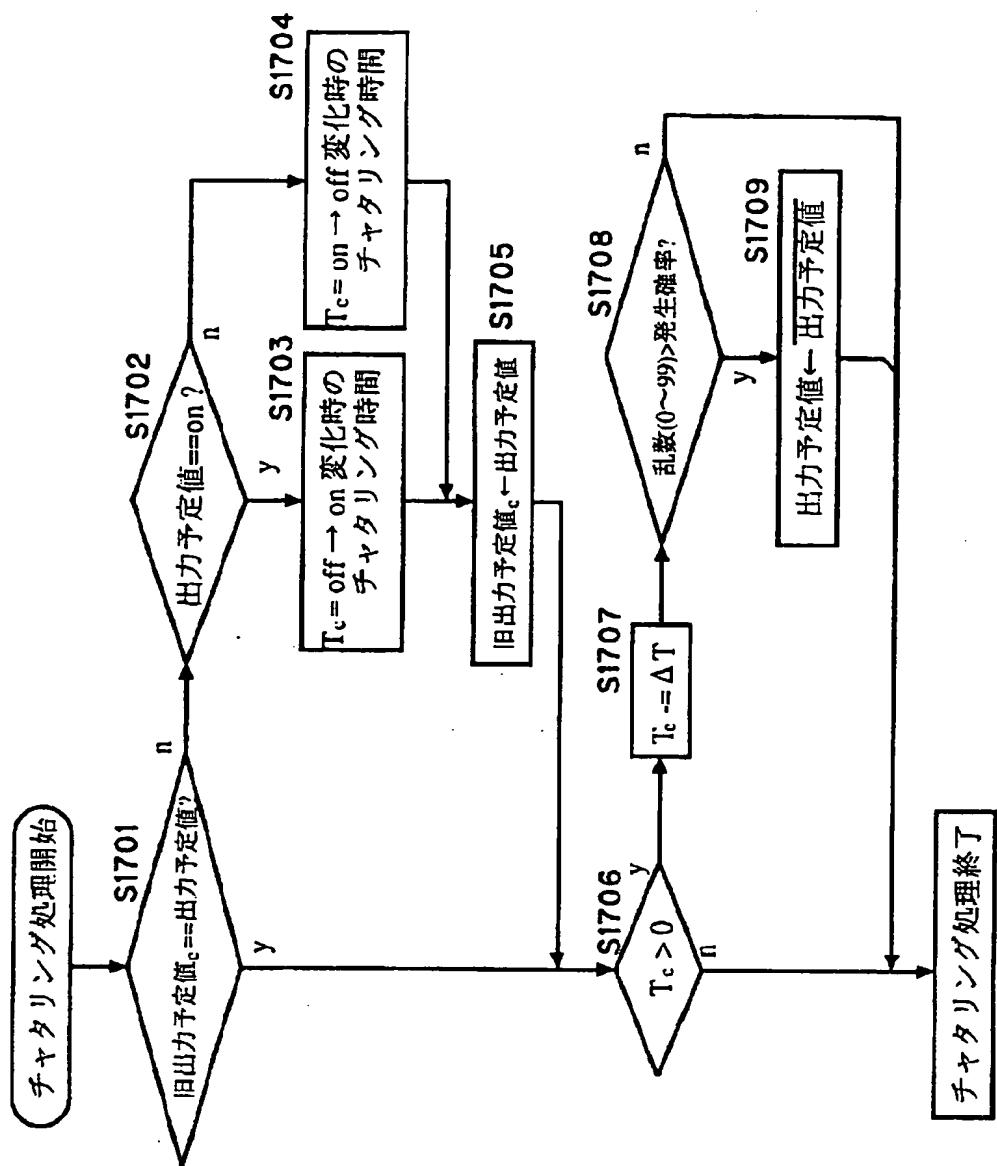
【図40】



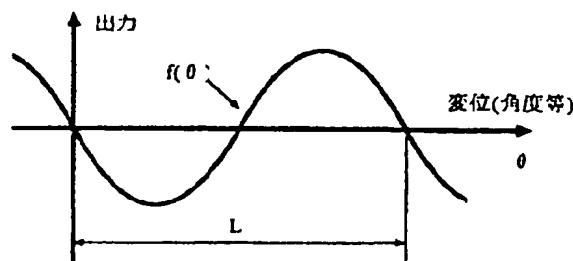
【図4-1】



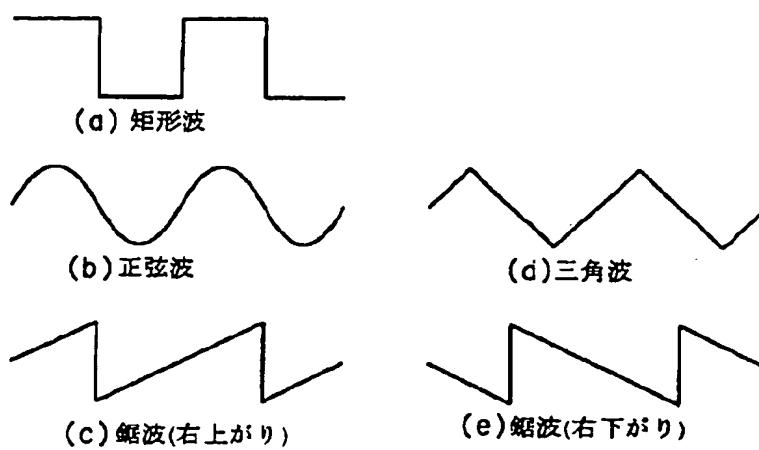
【図4-2】



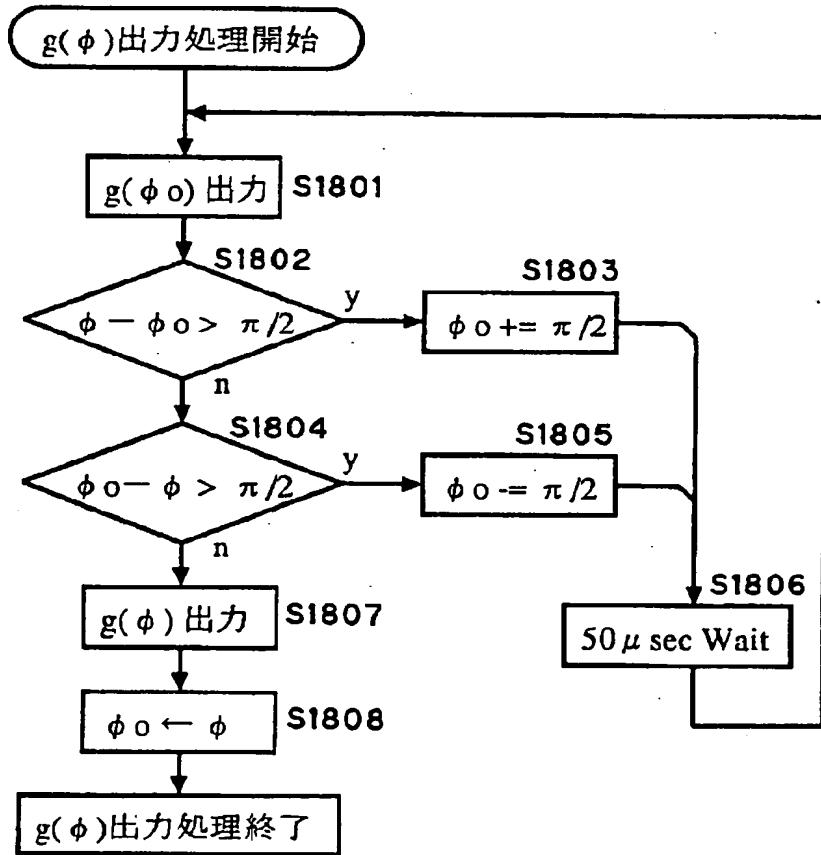
【図4-3】



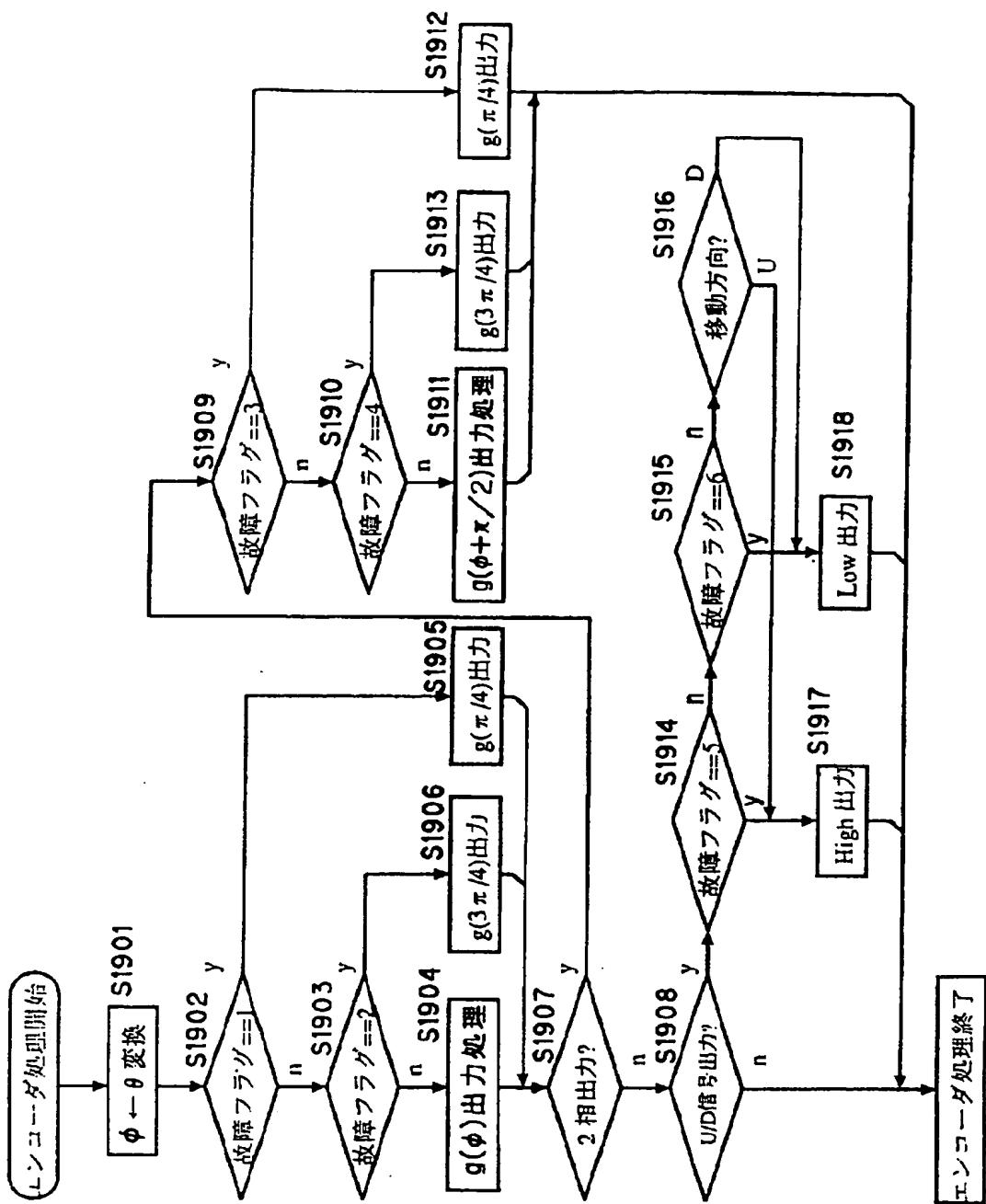
【図44】



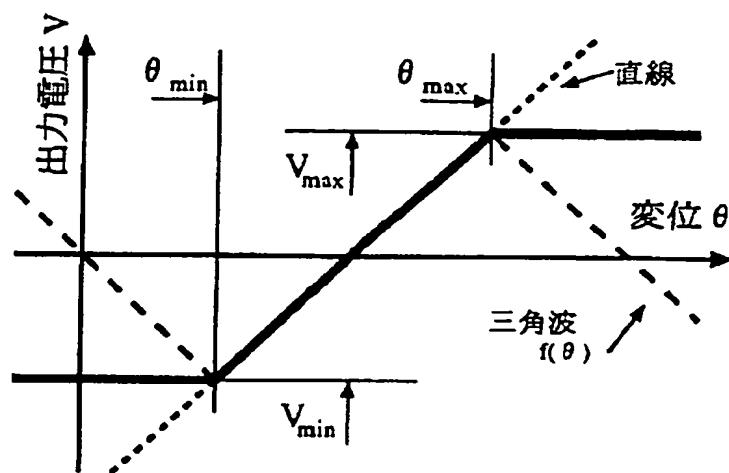
【図45】



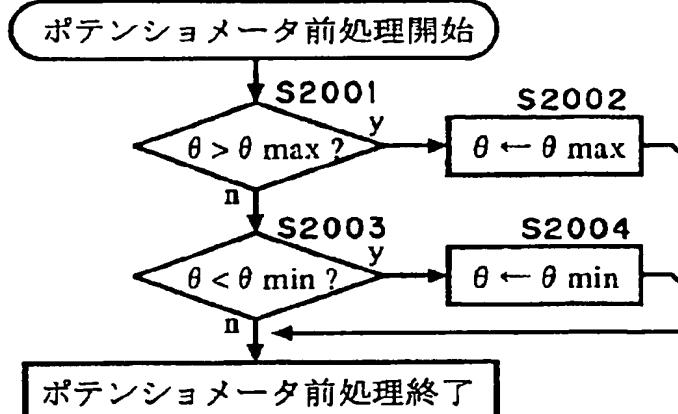
【図46】



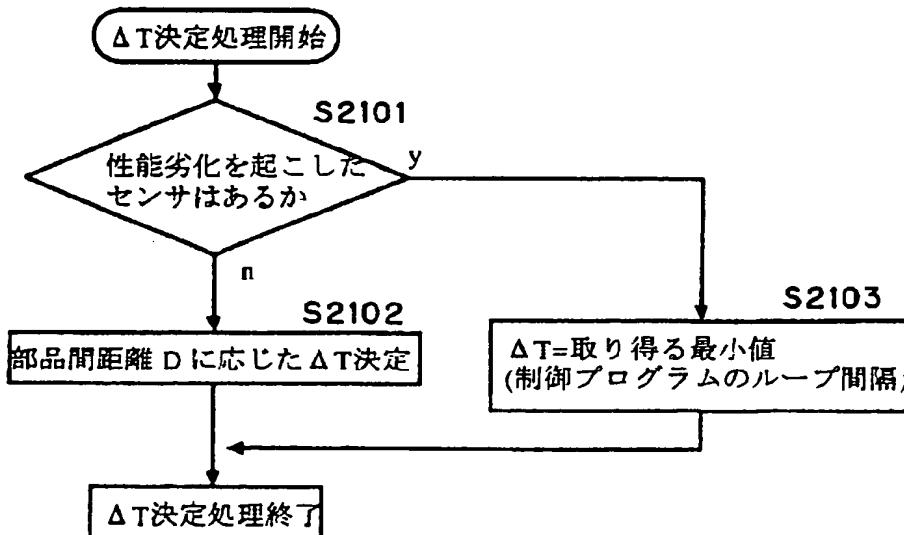
【図47】



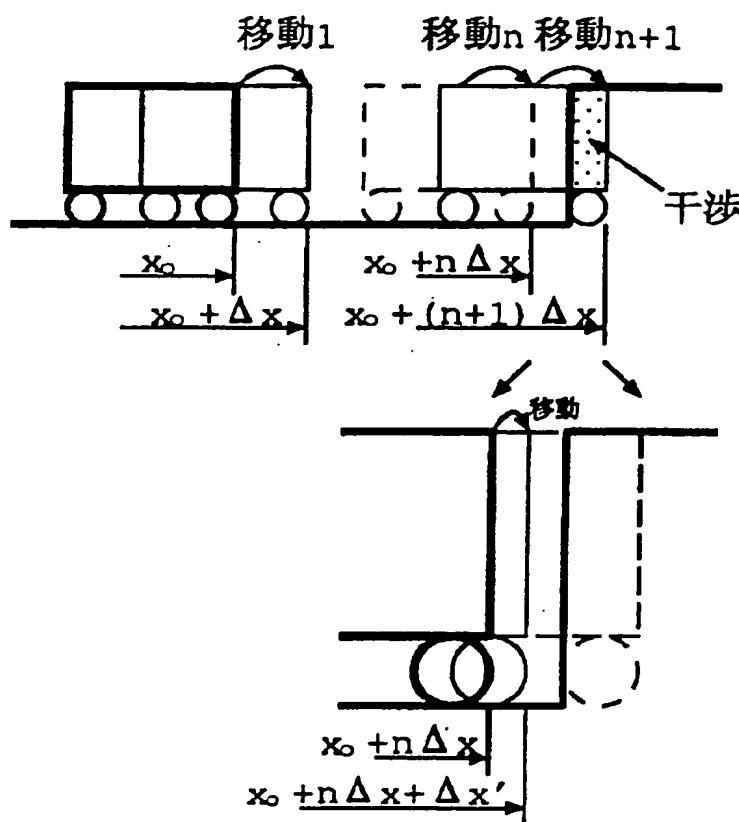
【図48】



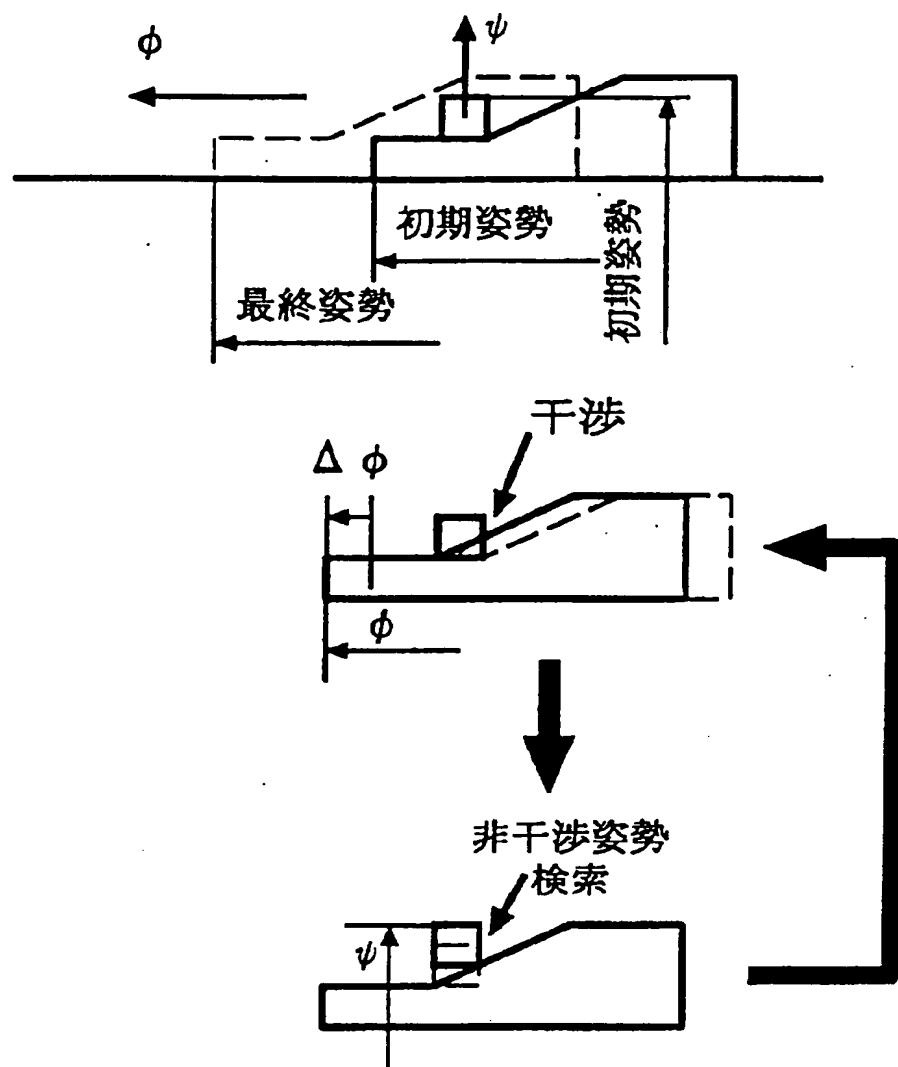
【図49】



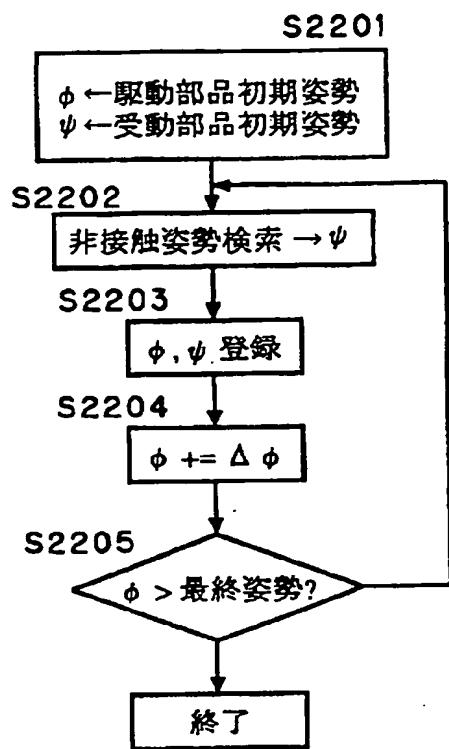
【図50】



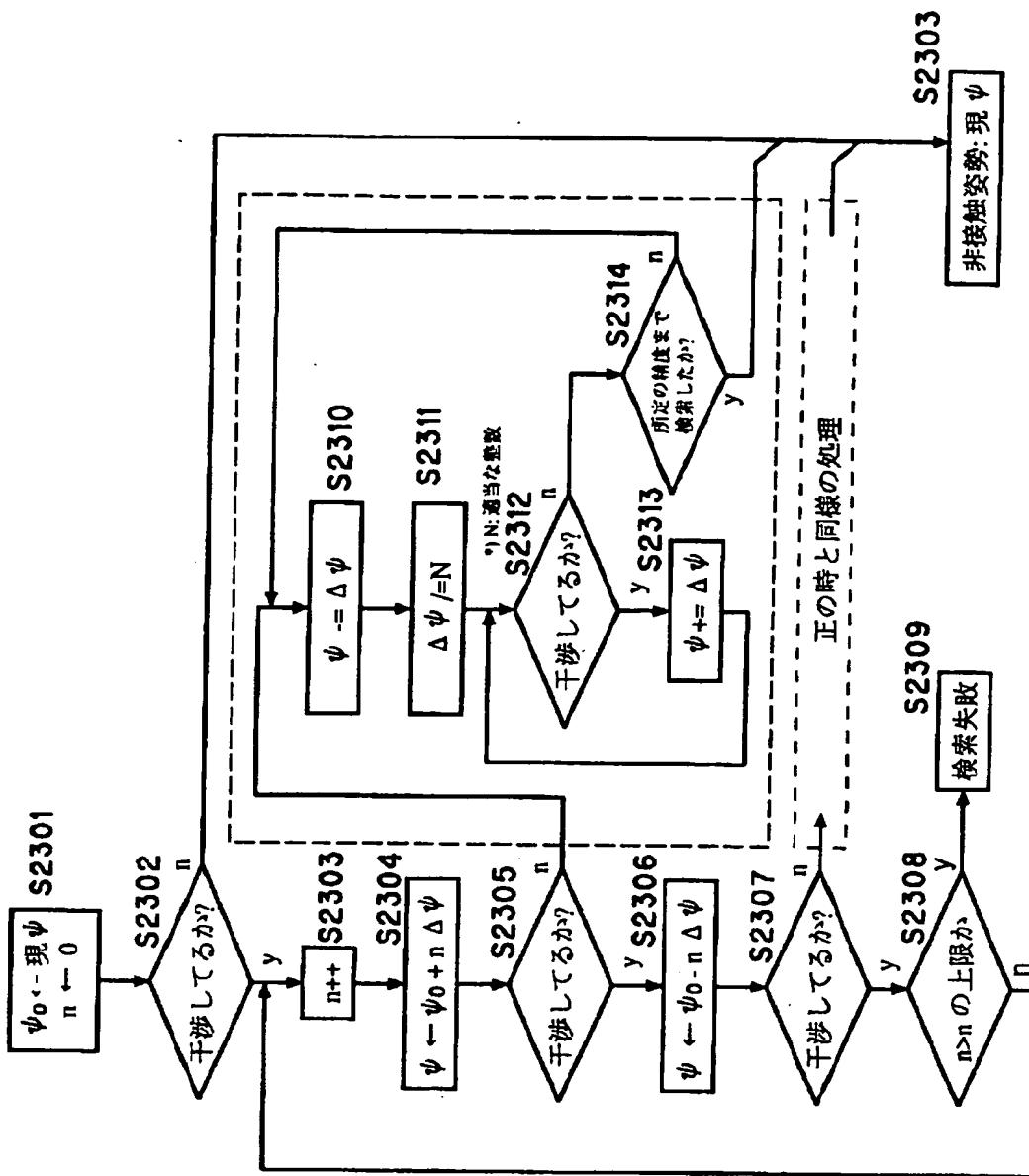
【図5.1】



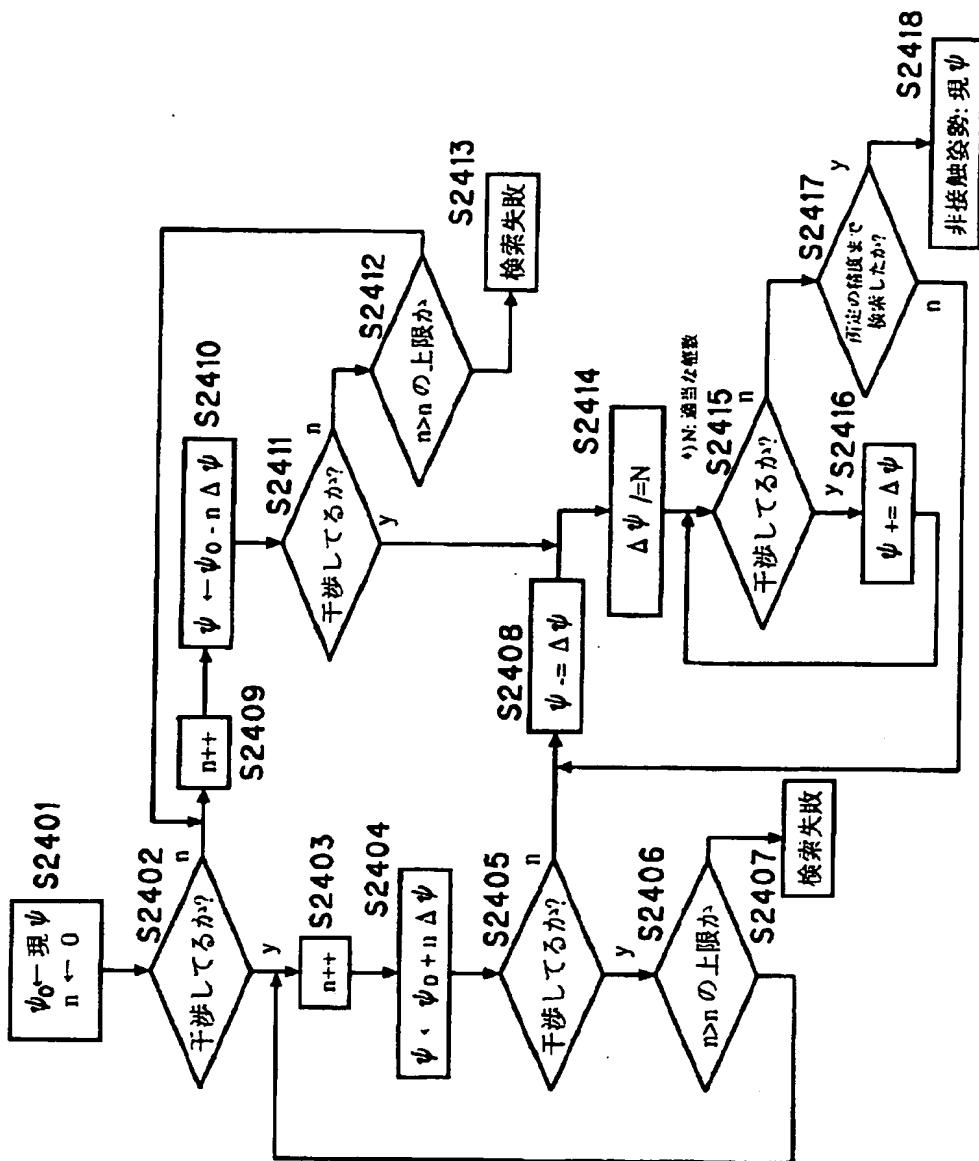
【図52】



【図53】



【図 5.4】



【図 5 5】

